Algorithmica (2011) 60: 653–678
DOI 10.1007/s00453-009-9366-4

Knowledge State Algorithms

Wolfgang Bein · Lawrence L. Larmore · John Noga · Rüdiger Reischuk

Received: 2 December 2008 / Accepted: 1 October 2009 / Published online: 17 October 2009

Abstract We introduce the novel concept of knowledge states. The knowledge state approach can be used to construct competitive randomized online algorithms and study the trade-off between competitiveness and memory. Many well-known algorithms can be viewed as knowledge state algorithms. A knowledge state consists of a distribution of states for the algorithm, together with a work function which approximates the conditional obligations of the adversary. When a knowledge state algorithm receives a request, it then calculates one or more "subsequent" knowledge states, together with a probability of transition to each. The algorithm uses randomization to select one of those subsequents to be the new knowledge state. We apply this method to randomized k-paging. The optimal minimum competitiveness of any randomized online algorithm for the k-paging problem is the kth harmonic number, $H_k = \sum_{i=1}^{k} \frac{1}{i}$. Existing algorithms which achieve that optimal competitiveness must keep bookmarks, *i.e.*, memory of the names of pages not in the cache. An H_k-competitive randomized algorithm for that problem which uses $O(k)$ bookmarks is

Research of W. Bein supported by NSF grant CCR-0312093.
Research of L.L. Larmore supported by NSF grant CCR-0312093.

W. Bein (✉) · L.L. Larmore
Center for the Advanced Study of Algorithms, School of Computer Science, University of Nevada, Las Vegas, NV 89154, USA
e-mail: bein@cs.unlv.edu

L.L. Larmore
e-mail: larmore@cs.unlv.edu

J. Noga
Department of Computer Science, California State University, Northridge, CA 91330, USA
e-mail: jnoga@csun.edu

R. Reischuk
Institut für Theoretische Informatik, Universität Lübeck, Ratzeburger Allee 160, 23538 Lübeck, Germany
e-mail: reischuk@tcs.uni-luebeck.de

presented, settling an open question by Borodin and El-Yaniv. In the special cases where $k = 2$ and $k = 3$, solutions are given using only one and two bookmarks, respectively.

Keywords Competitive analysis · Online algorithms · Task systems · Randomization · Paging · Server problem

1 Motivation and Background

In this paper we introduce a new method for constructing randomized online algorithms, which we call the *knowledge state* method. The purpose of this method is to address the trade-off between memory and competitiveness. A *knowledge state* gives a distribution for the algorithm, while at the same time approximating conditional obligations of an adversary with a work function. When a knowledge state algorithm receives a request, it calculates one or more "subsequent" knowledge states, together with a probability of transition to each. It then moves to a new knowledge state, possibly "forgetting" information, and thus saving space. Fundamentally, the knowledge states enable the algorithm to remember limited information while still achieving competitiveness.

The model is introduced and fully described for the first time in this publication, but we note that some published algorithms can be seen to be knowledge state algorithms. For example, the algorithm EQUITABLE [1] is a knowledge state algorithm for the k-paging problem that achieves the optimal randomized competitiveness of H_k for each k, using only $O(k^2 \log k)$ memory, as opposed to the prior algorithm, PARTITION [14], which requires unlimited memory as the length of the request sequence grows. At the other end of the scale, the randomized algorithm RANDOM_SLACK [12] is an extremely simple knowledge state algorithm which achieves randomized 2-competitiveness for the 2-server problem for all metric spaces, and which achieves randomized k-competitiveness for the k-server problem on some spaces, including trees. We also note that RANDOM_SLACK is *trackless*, meaning that the algorithm does not keep "track" of any point where it does not have a server. (See the ACM SIGACT column [6] for a summary of tracklessness; see also [3–5, 7].)

There appears to be a trade-off between competitiveness and memory for randomized online paging algorithms. The randomized k-paging algorithm EQUITABLE given by Achlioptas *et al.* [1] is H_k-competitive and uses $O(k^2 \log k)$ memory. This competitive ratio is best possible. The randomized algorithm RMARK given by Fiat *et al.* [13] is $(2H_k - 1)$-competitive, but only uses $O(k)$ memory. Borodin and El Yaniv [9] list as an open question whether there exists an H_k-competitive randomized algorithm which requires $O(k)$ memory for k-paging. In this paper we answer this question in the affirmative.

Chrobak, Koutsoupias and Noga [10] claim that, "From a purely practical standpoint, non-trackless algorithms are of limited interest as cache replacement strategies, as they cannot be realistically implemented." Unfortunately, the best competitiveness known for a randomized trackless algorithm for general k is $2H_k - 1$, achieved by RMARK [13]. Bein and Larmore [5] have shown that it is not possible for a trackless algorithm to achieve H_k-competitiveness if $k = 2$, and we expect that result to generalize to higher k. The algorithms in this paper are the result of an effort to provide optimally competitive randomized algorithms "as close as possible" to trackless.

We give a formal description of the knowledge state method for randomized online algorithms. It is defined using the *mixed model* of online computation, described in Sect. 2. In Sect. 2 we also relate the mixed model to the standard models of online computation, and explain how a behavioral algorithm can be derived from a mixed model description. Section 3 defines the knowledge state method using the mixed model. Section 4 gives results for the paging problem. We start with the case $k = 2$ to illustrate our method. The algorithm for $k = 2$ is optimally competitive and uses provably the smallest amount of memory. We then give a similar result for $k = 3$. Next the H_k-competitive randomized algorithm with $O(k)$ memory is given. Section 5 summarizes knowledge state results for the 2-server problem in Cross Polytope Spaces and for the caching problem in shared memory multiprocessor systems.

2 The Mixed Model of Online Computation

We introduce a new model of randomized online computation which is a generalization of both the classic behavioral and distributional models. We assume that we are given an online problem with states $\mathcal{X}$ (also called configurations), a fixed *start state* $x^0 \in \mathcal{X}$, and a set of requests, $\mathcal{R}$. If the current state is $x \in \mathcal{X}$ and a request $r \in \mathcal{R}$ is given, an algorithm for the problem must *service* the request by choosing a new state y and paying a cost, which we denote $\mathit{cost}(x, r, y)$. We assume that there is a "distance" function d on $\mathcal{X}$, and that it is possible to choose to move from state x to state y at cost $d(x, y)$ at any time, given any request, or no request. We further assume that $d(x, x) = 0$ and $d(x, z) \le d(x, y) + d(y, z)$ for any states x, y, z, and that $\mathit{cost}(u, r, v) \le d(u, x) + \mathit{cost}(x, r, y) + d(y, v)$ for any states u, x, y, v and request r. Examples of online problems satisfying these conditions abound, such as the server problem, the paging problem, and the CNN problem [9].

Given a *request sequence* $\varrho = r^1, \dots, r^n$, an algorithm must choose a sequence of states $x^1, \dots, x^n$, called the *service sequence*. The *cost* of this service sequence is defined to be $\sum_{t=1}^{n} \mathit{cost}(x^{t-1}, r^t, x^t)$. An *offline* algorithm knows ϱ before choosing the service sequence, while an *online* algorithm must choose x^t without knowledge of future requests. We will assume that there is an optimal offline algorithm, *opt*, which computes an optimal service sequence for any given request sequence. As is customary we say that a deterministic online algorithm $\mathcal{A}$ is C*-competitive* for a given number C if there exists a constant K (not dependent on ϱ) such that $\mathit{cost}_{\mathcal{A}}(\varrho) \le C \cdot \mathit{cost}_{\mathit{opt}}(\varrho) + K$ for any request sequence ϱ. Similarly, we say that a randomized online algorithm $\mathcal{A}$ is C-competitive for a given number C if there exists a constant K (not dependent on ϱ) such that $\mathit{Exp}(\mathit{cost}_{\mathcal{A}}(\varrho)) \le C \cdot \mathit{cost}_{\mathit{opt}}(\varrho) + K$ for any request sequence ϱ, where *Exp* denotes expected value.

In order to make the description of various models of randomized online computation more precise, we introduce the following notation. Let Π be the set of all finite distributions on $\mathcal{X}$. If $\pi \in \Pi$ and $\mathcal{S} \subseteq \mathcal{X}$, we say that $\mathcal{S}$ *supports* the distribution π if $\pi(\mathcal{S}) = 1$. The support of any $\pi \in \Pi$ is defined to be the unique minimal set which supports π. By abuse of notation, if the support of π is a singleton $\{x\}$, we write $\pi = x$.

An instance of the *transportation problem* is a weighted directed bipartite graph with distributions on both parts. Formally, an instance is an ordered quintuple $(A, B, cost, \alpha, \beta)$ where A and B are finite non-empty sets, α is a distribution on A, β is a distribution on B, and *cost* is a real-valued function on $A \times B$. A *solution* to that instance is a distribution γ on $A \times B$ such that

1. $\gamma(\{a\} \times B) = \alpha(a)$ for all $a \in A$.
2. $\gamma(A \times \{b\}) = \beta(b)$ for all $b \in B$.

Then $cost(\gamma) = \sum_{a\in A}\sum_{b\in B}\gamma(a,b)cost(a,b)$, and γ is a *minimal* solution if $cost(\gamma)$ is minimized over all solutions, in which case we call $cost(\gamma)$ the *minimum transportation cost*.

There are three standard models of randomized online algorithms (see, for example [9]). We introduce a new model in this paper, which we call the *mixed model*. Those three standard models are: distribution of deterministic online algorithms, the behavioral model, and the distributional model. We very briefly describe the three standard models.

Distribution of Deterministic Online Algorithms In this model, $\mathcal{A}$ is a random variable whose value is a deterministic online algorithm. If the random variable has a finite distribution, we say that $\mathcal{A}$ is *barely random*.

Behavioral Online Algorithms In this model $\mathcal{A}$ uses randomization at each step to pick the next configuration. We assume that $\mathcal{A}$ has memory. Let $\mathcal{M}$ be the set of all possible memory states of $\mathcal{A}$. We define a *full state* of $\mathcal{A}$ to be an ordered pair $k = (x, \mu) \in \mathcal{X} \times \mathcal{M}$. Let $\mu^0 \in \mathcal{M}$ be the initial memory state, and let μ^t be the memory state of $\mathcal{A}$ after servicing the first t requests.

Then $\mathcal{A}$ uses randomization to compute $k^t = (x^t, \mu^t)$, the full state after t steps, given only k^{t-1} and r^t. A behavioral algorithm can then be thought of as a function on $\mathcal{X} \times \mathcal{M} \times \mathcal{R}$ whose values are random variables in $\mathcal{X} \times \mathcal{M}$.

Distributional Online Algorithms If $\pi, \pi' \in \Pi$, let $\mathcal{S}$ be the support of π and $\mathcal{S}'$ be the support of π'. If $r \in \mathcal{R}$, we define $cost(\pi, r, \pi')$ to be the minimum transportation cost of the transportation problem $(\mathcal{S}, \mathcal{S}', cost^r, \pi, \pi')$, where $cost^r = cost(\ ,r,\ ) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

A distributional online algorithm $\mathcal{A}$ is then defined as follows.

1. There is a set $\mathcal{M}$ of memory states of $\mathcal{A}$. There is a start memory state $\mu^0 \in \mathcal{M}$.
2. A *full state* of $\mathcal{A}$ is a pair $k = (\pi, \mu) \in \Pi \times \mathcal{M}$. The initial full state is $k^0 = (\pi^0, \mu^0)$, where $\pi^0 = x^0$.
3. For any given full state $k = (\pi, \mu)$ and request r, $\mathcal{A}$ deterministically computes a new full state $k' = (\pi', \mu')$, using only the inputs π, μ, and r. We write $\mathcal{A}(\pi, \mu, r) = (\pi', \mu')$ or alternatively $\mathcal{A}(k, r) = k'$. Thus, $\mathcal{A}$ is a function from $\Pi \times \mathcal{M} \times \mathcal{R}$ to $\Pi \times \mathcal{M}$.
4. Given any input sequence $\varrho = r^1, \ldots, r^n$, $\mathcal{A}$ computes a sequence of full states $\mathcal{A}(\varrho) = k^1, \ldots, k^n$, following the rule that $k^t = (\pi^t, \mu^t) = \mathcal{A}(k^{t-1}, r^t)$ for all $t \geq 1$. Define $cost_{\mathcal{A}}(\varrho) = \sum_{t=1}^n cost(\pi^{t-1}, r^t, \pi^t)$.

We note that a distributional online algorithm, despite being a model for a randomized online algorithm, is in fact deterministic, in the sense that the full states $\{k^t\}$ are computed deterministically.

The following theorem is well-known. (It is, for example, implicit in Chap. 6 of [9].)

Theorem 1 *All three of the above models of randomized online algorithms are equivalent, in the following sense. If $\mathcal{A}_1$ is an algorithm of one of the models, there exist algorithms $\mathcal{A}_2, \mathcal{A}_3$, of each of the other models, such that, given any request sequence ϱ, the cost (or expected cost) of each $\mathcal{A}_i$ for ϱ is no greater than the cost (or expected cost) of $\mathcal{A}_1$.*

The Mixed Model The *mixed model* of randomized algorithms is a generalization of both the behavioral model and the distributional model. A mixed online algorithm chooses a distribution at each step, but, as opposed to a distributional algorithm, which must make that choice deterministically, can use randomization to choose the distribution.

A *mixed* online algorithm $\mathcal{A}$ for an online problem $\mathcal{P} = (\mathcal{X}, \mathcal{R}, d)$ is defined as follows. As before, let Π be the set of finite distributions on $\mathcal{X}$.

1. There is a set $\mathcal{M}$ of memory states of $\mathcal{A}$. There is a start memory state $\mu^0 \in \mathcal{M}$.
2. A *full state* of $\mathcal{A}$ is a pair $k = (\pi, \mu) \in \Pi \times \mathcal{M}$. The initial full state is $k^0 = (\pi^0, \mu^0)$, where $\pi^0 = x^0$.
3. For any given full state $k = (\pi, \mu)$ and request r, there exists a finite set of full states $k_1, \dots, k_m$ and probabilities $\lambda_1, \dots, \lambda_m$, where $\sum_{i=1}^m \lambda_i = 1$, such that if the current full state is k and the next request is r, $\mathcal{A}$ uses randomization to compute a new full state $k' = (\pi', \mu')$, by selecting $k' = k_i$ for some i. The probability that $\mathcal{A}$ selects each given k_i is λ_i. We call the $\{k_i\}$ the *subsequents* and the $\{\lambda_i\}$ the *weights* of the subsequents, for the request r from the full state k.
   $\mathcal{A}$ is a function on $\Pi \times \mathcal{M} \times \mathcal{R}$ whose values are random variables in $\Pi \times \mathcal{M}$. We can write $\mathcal{A}(\pi, \mu, r) = (\pi', \mu')$. Alternatively, we write $\mathcal{A}(k, r) = k'$. For fixed k and r, we regard k', π', and μ' as random variables.
4. Given any input sequence $\varrho = r^1, \dots, r^n$, $\mathcal{A}$ computes a sequence of full states $\mathcal{A}(\varrho) = (\pi^1, \mu^1), \dots, (\pi^n, \mu^n)$, following the rule that $k^t = (\pi^t, \mu^t) = \mathcal{A}(k^{t-1}, r^t)$ for all $t > 1$. Note that, for all $t > 0$, k^t, π^t, and μ^t are random variables.

Computing the cost of a step of a mixed model online algorithm $\mathcal{A}$ is somewhat tricky. We note that it might seem that $\sum_{i=1}^m \lambda_i cost(\pi, r, \pi_i)$ would be that cost; however, this is an overestimate.

Let $k = (\pi, \mu) \in \Pi \times \mathcal{M}$ and let $r \in \mathcal{R}$. Let $\mathcal{S} \subseteq \mathcal{X}$ be the support of π. Let $\{k_i = (\pi_i, \mu_i)\}$ be the subsequents and $\{\lambda_i\}$ the weights of the subsequents, for the request r from the full state k. Let $\bar{\mathcal{S}} \subseteq \mathcal{X}$ be the union of the supports of the $\{\pi_i\}$. Define $\bar{\pi} = \sum_{i=1}^m \lambda_i \pi_i$. Note that $\bar{\pi} \in \Pi$, and its support is $\bar{\mathcal{S}}$. Define $cost_{\mathcal{A}}(k, r) = cost(\pi, r, \bar{\pi})$.

Finally, if $\varrho = r^1, \dots, r^n$ is the input request sequence, and the sequence of full states of $\mathcal{A}$ is $k^1, \dots, k^n$, we define $cost_{\mathcal{A}}(\varrho) = \sum_{t=1}^n cost_{\mathcal{A}}(k^{t-1}, r^t)$.

We now prove that the mixed model for randomized online algorithms is equivalent to the three standard models.

Lemma 1 *If $\mathcal{A}$ is a mixed online algorithm, there is a behavioral online algorithm $\tilde{\mathcal{A}}$ such that, for any request sequence ϱ, $Exp(cost_{\tilde{\mathcal{A}}}(\varrho)) = Exp(cost_{\mathcal{A}}(\varrho))$, and such that the space complexity of $\tilde{\mathcal{A}}$ is asymptotically the same as the space complexity of $\mathcal{A}$.*

Proof A memory state of $\tilde{\mathcal{A}}$ will be a full state of $\mathcal{A}$, *i.e.*, we could write $\tilde{\mathcal{M}} \subseteq \Pi \times \mathcal{M}$. By a slight abuse of notation, we also define a full state of $\tilde{\mathcal{A}}$ to be an ordered triple $(x, \pi, \mu) \in \mathcal{X} \times \Pi \times \mathcal{M}$ such that (π, μ) is a full state of $\mathcal{A}$ and $\pi(x) > 0$. Intuitively, $\tilde{\mathcal{A}}$ keeps track of its true state $x \in \mathcal{X}$, while remembering the full state (π, μ) of an emulation of $\mathcal{A}$.

For clarity of the proof, we introduce more complex notation for some of the quantities defined earlier. Let $\pi, \sigma \in \Pi$, $\mu, \nu \in \mathcal{M}$, and $r \in \mathcal{R}$. If (π, μ) is a full state of $\mathcal{A}$, define $\lambda_{\pi,\mu,r,\sigma,\nu}$ to be the probability that $\mathcal{A}(\pi, \mu, r) = (\sigma, \nu)$, *i.e.*, the conditional probability that $\mathcal{A}$ chooses (σ, ν) to be the next full state, given that the current full state is (π, μ) and the request is r. We assume that there can be at most finitely many choices of (σ, ν) for which $\lambda_{\pi,\mu,r,\sigma,\nu} > 0$. In case (π, μ) is not a full state of $\mathcal{A}$, then $\lambda_{\pi,\mu,r,\sigma,\nu}$ is defined to be zero. If (π, μ) is a full state of $\mathcal{A}$ and $r \in \mathcal{R}$, write $\bar{\pi}_{\pi,\mu,r} = \sum_{\sigma\in\Pi,\nu\in\mathcal{M}} \lambda_{\pi,\mu,r,\sigma,\nu} \cdot \sigma \in \Pi$, and choose a finite distribution $\gamma_{\pi,\mu,r}$ on $\mathcal{X} \times \mathcal{X}$ which is a minimal solution to the transportation problem $(\mathcal{X}, \mathcal{X}, cost^r, \pi, \bar{\pi}_{\pi,\mu,r})$, where $cost^r(x, y) = cost(x, r, y)$. Thus $\pi(x) = \sum_{y\in\mathcal{X}} \gamma_{\pi,\mu,r}(x, y)$ for $x \in \mathcal{X}$; $\bar{\pi}_{\pi,\mu,r}(y) = \sum_{x\in\mathcal{X}} \gamma_{\pi,\mu,r}(x, y)$ for $y \in \mathcal{X}$; and $cost_{\mathcal{A}}(\pi, \mu, r) = \sum_{x\in\mathcal{X},y\in\mathcal{X}} \gamma_{\pi,\mu,r}(x, y) cost(x, r, y)$.

We now formally describe the action of the behavioral algorithm $\tilde{\mathcal{A}}$. The initial full state of $\tilde{\mathcal{A}}$ is $(x^0, k^0) = (x^0, \pi^0, \mu^0)$. Given that the full state of $\tilde{\mathcal{A}}$ is (x, π, μ) and the next request is $r \in \mathcal{R}$, and given any $(y, \sigma, \nu) \in \mathcal{X} \times \Pi \times \mathcal{M}$, we define $\Lambda_{x,\pi,\mu,r,y,\sigma,\nu}$, the probability that $\tilde{\mathcal{A}}$ chooses the next full state to be (y, σ, ν), as follows:

If $\bar{\pi}_{\pi,\mu,r}(y) = 0$, then $\Lambda_{x,\pi,\mu,r,y,\sigma,\nu} = 0$.
Otherwise, $\Lambda_{x,\pi,\mu,r,y,\sigma,\nu} = \frac{\gamma_{\pi,\mu,r}(x,y)\cdot\sigma(y)\cdot\lambda_{\pi,\mu,r,\sigma,\nu}}{\pi(x)\cdot\bar{\pi}_{\pi,\mu,r}(y)}$.

Let ϱ be a given request sequence. We now prove that $Exp(cost_{\tilde{\mathcal{A}}}(\varrho)) = Exp(cost_{\mathcal{A}}(\varrho))$. For any $t \geq 0$ and any full state (π, μ) of $\mathcal{A}$, define $p^t(\pi, \mu)$ to be the probability that the full state of $\mathcal{A}$ is (π, μ) after t steps. Additionally, if $x \in \mathcal{X}$, define $q^t(x, \pi, \mu)$ to be the probability that the full state of $\tilde{\mathcal{A}}$ is (x, π, μ) after t steps.

To prove the lemma we consider first the following two claims:

Claim 1 *For any $t \geq 0$, $x \in \mathcal{X}$, $\pi \in \Pi$, and $\mu \in \mathcal{M}$, $q^t(x, \pi, \mu) = p^t(\pi, \mu) \cdot \pi(x)$.*

Claim 2 *For any $t \geq 0$, $\pi \in \Pi$, and $\mu \in \mathcal{M}$, $\sum_{x\in\mathcal{X}} q^t(x, \pi, \mu) = p^t(\pi, \mu)$.*

We prove Claims 1 and 2 by simultaneous induction on t. If $t = 0$, both claims are trivial by definition. Now, suppose $t > 0$. We verify Claim 1 for t. By the inductive hypothesis, Claim 2 holds for $t-1$. Write $r=r^t$. Let $y,\sigma,\nu\in\mathcal{X}\times\Pi\times\mathcal{M}$. If (σ,ν) is not a full state of $\mathcal{A}$ or $\sigma(y)=0$, we are done. Otherwise, recall that $\bar{\pi}_{\pi,\mu,r}(y)=\sum_{x\in\mathcal{X}}\gamma_{\pi,\mu,r}(x,y)$ for all $y\in\mathcal{X}$, and we obtain

$$\begin{aligned}
q^t(y,\sigma,\nu) &= \sum_{(x,\pi,\mu)\in\mathcal{X}\times\Pi\times\mathcal{M}} q^{t-1}(x,\pi,\mu)\Lambda_{x,\pi,\mu,r,y,\sigma,\nu}\\
&= \sum_{(x,\pi,\mu)\in\mathcal{X}\times\Pi\times\mathcal{M},\pi(x)>0,\bar{\pi}_{\pi,\mu,r}(y)>0} p^{t-1}(\pi,\mu)\pi(x)\\
&\quad\cdot\frac{\gamma_{\pi,\mu,r}(x,y)\cdot\sigma(y)\cdot\lambda_{\pi,\mu,r,\sigma,\nu}}{\pi(x)\cdot\bar{\pi}_{\pi,\mu,r}(y)}\\
&= \sum_{(x,\pi,\mu)\in\mathcal{X}\times\Pi\times\mathcal{M},\bar{\pi}_{\pi,\mu,r}(y)>0} p^{t-1}(\pi,\mu)\cdot\frac{\gamma_{\pi,\mu,r}(x,y)\cdot\sigma(y)\cdot\lambda_{\pi,\mu,r,\sigma,\nu}}{\bar{\pi}_{\pi,\mu,r}(y)}\\
&= \sigma(y)\cdot\sum_{(\pi,\mu)\in\Pi\times\mathcal{M},\bar{\pi}_{\pi,\mu,r}(y)>0}\Bigg(p^{t-1}(\pi,\mu)\cdot\lambda_{\pi,\mu,r,\sigma,\nu}\\
&\quad\cdot\sum_{x\in\mathcal{X}}\frac{\gamma_{\pi,\mu,r}(x,y)}{\bar{\pi}_{\pi,\mu,r}(y)}\Bigg)\\
&= \sigma(y)\cdot\sum_{(\pi,\mu)\in\Pi\times\mathcal{M},\bar{\pi}_{\pi,\mu,r}(y)>0} p^{t-1}(\pi,\mu)\cdot\lambda_{\pi,\mu,r,\sigma,\nu}\\
&= \sigma(y)\cdot\sum_{(\pi,\mu)\in\Pi\times\mathcal{M}} p^{t-1}(\pi,\mu)\cdot\lambda_{\pi,\mu,r,\sigma,\nu}=\sigma(y)\cdot p^t(\sigma,\nu)
\end{aligned}$$

which verifies Claim 1 for t. Claim 2 for t follows trivially.

For the conclusion of the lemma, let $t>0$, and let $r=r^t$. We use Claim 1 for $t-1$. Recall that $\bar{\pi}_{\pi,\mu,r}=\sum_{\sigma\in\Pi,\nu\in\mathcal{M}}\lambda(\pi,\mu,r,\sigma,\nu)\cdot\sigma$ for any full state (π,μ) of $\mathcal{A}$. Then

$$\begin{aligned}
Exp\Big(cost^t_{\tilde{\mathcal{A}}}\Big) &= \sum_{\pi,\sigma\in\Pi,\mu,\nu\in\mathcal{M},x,y\in\mathcal{X}} q^{t-1}(x,\pi,\mu)\cdot\Lambda_{x,\pi,\mu,r,y,\sigma,\nu}\cdot cost(x,r,y)\\
&= \sum_{\substack{\pi,\sigma\in\Pi,\mu,\nu\in\mathcal{M},x,y\in\mathcal{X}\\ \pi(x)>0,\sigma(y)>0}} p^{t-1}(\pi,\mu)\cdot\pi(x)\\
&\quad\cdot\frac{\gamma_{\pi,\mu,r}(x,y)\cdot\sigma(y)\cdot\lambda_{\pi,\mu,r,\sigma,\nu}}{\pi(x)\cdot\bar{\pi}_{\pi,\mu,r}(y)}\cdot cost(x,r,y)\\
&= \sum_{\pi\in\Pi,\mu\in\mathcal{M},x,y\in\mathcal{X}}\Bigg(p^{t-1}(\pi,\mu)\cdot\gamma_{\pi,\mu,r}(x,y)\cdot cost(x,r,y)\\
&\quad\cdot\sum_{\sigma\in\Pi,\nu\in\mathcal{M},\sigma(y)>0}\frac{\lambda_{\pi,\mu,r,\sigma,\nu}\cdot\sigma(y)}{\bar{\pi}_{\pi,\mu,r}(y)}\Bigg)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{\pi\in\Pi,\mu\in\mathcal{M},x,y\in\mathcal{X}} p^{t-1}(\pi,\mu)\cdot\gamma_{\pi,\mu,r}(x,y)\cdot cost(x,r,y)\\
&= \sum_{\pi\in\Pi,\mu\in\mathcal{M}} \left(p^{t-1}(\pi,\mu)\cdot\sum_{x,y\in\mathcal{X}}\gamma_{\pi,\mu,r}(x,y)\cdot cost(x,r,y)\right)\\
&= \sum_{\pi\in\Pi,\mu\in\mathcal{M}} p^{t-1}(\pi,\mu)\cdot cost_{\mathcal{A}}(\pi,r,\bar{\pi}_{\pi,\mu,r})\\
&= \sum_{\pi\in\Pi,\mu\in\mathcal{M}} p^{t-1}(\pi,\mu)\cdot cost_{\mathcal{A}}(\pi,\mu,r) = Exp\big(cost^t_{\mathcal{A}}\big).
\end{aligned}
$$

Finally, we note that $\mathcal{A}$ must remember not only its memory state μ, but also its distribution. Thus, the space complexity of $\tilde{\mathcal{A}}$ is asymptotically equal to the space complexity of $\mathcal{A}$. □

Theorem 2 *If $\mathcal{A}$ is a mixed model online algorithm for an online problem $\mathcal{P}$, there exist algorithms $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ for $\mathcal{P}$, of each of the standard models, such that, given any request sequence ϱ, the cost (or expected cost) of each $\mathcal{A}_i$ for ϱ is no greater than the cost (or expected cost) of $\mathcal{A}$.*

Proof From Lemma 1 and Theorem 1. □

Corollary 1 *If there is a C-competitive mixed model online algorithm for an online problem $\mathcal{P}$, there is a C-competitive online algorithm for $\mathcal{P}$ for each of the three standard models of randomized online algorithms.*

3 Knowledge State Algorithms

Let $\mathcal{X}$ be the set of configurations of an online problem. We say that a function $\omega : \mathcal{X} \to \mathbb{R}$ is *Lipschitz* if $\omega(y) \le \omega(x) + d(x,y)$ for all $x, y \in \mathcal{X}$. A *work function* is a non-negative Lipschitz function $\mathcal{X} \to \mathbb{R}$. If $\mathcal{S} \subseteq \mathcal{X}$, we say that $\mathcal{S}$ *supports* ω if, for any $y \in \mathcal{X}$ there exists some $x \in \mathcal{S}$ such that $\omega(y) = \omega(x) + d(x,y)$. If ω is supported by a finite set, then there is a unique minimal set $supp(\omega)$ which supports ω, which we call the *support* of ω. (All work functions considered in this paper have finite support.) If the minimum value of a work function is zero, we say it is an *offset function*. If ω is a work function, we define $\bar{\omega} = \omega - \min\omega$, the *offset* of ω.

The simplest example of a work function is a *cone*. If $x \in \mathcal{X}$, define $\chi_x(y) = d(x,y)$ for all $y \in \mathcal{X}$. Then χ_x, which we call the *cone* on x, has support $\{x\}$.

If ϱ is a request sequence, we define a work function ω^ϱ, which we call *the* work function of ϱ. If $x, y \in \mathcal{X}$, let $cost_{opt}(x,\varrho,y)$ be minimum cost of servicing the request sequence ϱ, starting at configuration x, and ending at configuration y. For any x, define $\omega^\varrho(x) = cost_{opt}(x^0,\varrho,x)$, where x^0 is the start configuration, and let $cost_{opt}(x^0,\varrho) = \min_x cost_{opt}(x^0,\varrho,x)$, the minimum cost of servicing ϱ. Similarly, the *offset function* of ϱ is defined to be $\bar{\omega}^\varrho = \omega^\varrho - cost_{opt}(x^0,\varrho)$. We will omit the parameter x^0 in our notation if the start state is understood.

We define an offset function ω to be *reachable* if there is some request sequence ϱ such that $\bar{\omega}^{\varrho} = \omega$.

Lemma 2 *Suppose ω and ω' are work functions, and $\mathcal{S}$ supports ω. Then $\omega(x) \geq \omega'(x)$ for all $x \in \mathcal{X}$ if and only if $\omega(y) \geq \omega'(y)$ for all $y \in \mathcal{S}$.*

Proof One direction of the proof is trivial. Suppose $\omega(y) \geq \omega'(y)$ for all $y \in \mathcal{S}$. Let $x \in \mathcal{X}$. There exists $y \in \mathcal{S}$ such that $\omega(x) = \omega(y) + d(y, x) \geq \omega'(y) + d(y, x) \geq \omega'(x)$. □

If ω is a work function and $r \in \mathcal{R}$ is a request, we define $(\omega \wedge r)(y) = \min_{x \in \mathcal{X}} \{\omega(x) + cost(x, r, y)\}$. We also define $\omega \overline{\wedge} r = \overline{\omega \wedge r}$. We refer to "$\wedge$" and "$\overline{\wedge}$" as *update* and *offset update*, respectively. The following lemma eases the computation of update in our applications.

Lemma 3 *If ω is supported by $\mathcal{S}$, then $(\omega \wedge r)(y) = \min_{x \in \mathcal{S}} \{\omega(x) + cost(x, r, y)\}$.*

Proof Trivially, $(\omega \wedge r)(y) \leq \min_{x \in \mathcal{S}} \{\omega(x) + cost(x, r, y)\}$. Pick $z \in \mathcal{X}$ such that $(\omega \wedge r)(y) = \omega(z) + cost(z, r, y)$. Pick $x \in \mathcal{S}$ such that $\omega(z) = \omega(x) + d(x, z)$. Then

$$\begin{aligned}(\omega \wedge r)(y) &= \omega(z) + cost(z, r, y) = \omega(x) + d(x, z) + cost(z, r, y)\\ &\geq \omega(x) + cost(x, r, y) \geq (\omega \wedge r)(y)\end{aligned}$$

and we are done. □

We can compute $\omega \wedge \varrho$ by repeated updates. The following lemma is well-known. (See, for example, [11].)

Lemma 4 *If $\varrho = r^1, \ldots, r^n$, let $\varrho^t = r^1, \ldots, r^t$ for all $t \leq n$. Then $\omega^0(x) = d(x^0, x)$ for all $x \in \mathcal{X}$ and $\omega^{\varrho^t} = \omega^{\varrho^{t-1}} \wedge r^t$ for all $t > 0$.*

Knowledge States A *knowledge state* is an ordered pair $k = (\pi, \omega)$, where π is a distribution on $\mathcal{X}$ and ω is a work function on $\mathcal{X}$.

We have used the word "support" in two different ways—the support of a work function, and the support of a distribution. Combining these two, we obtain a third meaning of the term: we say that $\mathcal{S} \subseteq \mathcal{X}$ supports the knowledge state (π, ω) if $\mathcal{S}$ supports π in the distribution sense, and also $\mathcal{S}$ supports ω in the work function sense. For each $x \in \mathcal{X}$, let $\kappa_x = (x, \chi_x)$ be the knowledge state supported by the singleton set $\{x\}$. We call κ_x the *cone knowledge state over* x.

A knowledge state algorithm $\mathcal{A}$ is a randomized algorithm in the mixed model, whose full state is a knowledge state. Formally:

1. There is a set of knowledge states $\mathcal{K} = \mathcal{K}_{\mathcal{A}}$, which includes the cone knowledge state κ_x for all $x \in \mathcal{X}$.
2. A *move* is defined to be an ordered pair $(k, r) \in \mathcal{K} \times \mathcal{R}$. For each move (k, r), there is a *subsequent vector* $\mathcal{A}(k, r) = (k_1, \ldots, k_m, \lambda_1, \ldots, \lambda_m)$, for some m (which may depend on k and r), such that

(a) The $\{\lambda_i\}$ are probabilities, *i.e.*, $\lambda_i \geq 0$ for all i, and $\sum \lambda_i = 1$.
(b) For each i, $k_i = (\pi_i, \omega_i) \in \mathcal{K}$.

We call the k_i the *subsequents* of the move, and we call λ_i the *weight* of the subsequent k_i.

We define the *algorithm cost* and the *estimated optimal cost* of the move (k, r) as follows. Recall that $k = (\pi, \omega)$, and $k_i = (\pi_i, \omega_i)$.

1. $cost_{\mathcal{A}}(k, r) = cost(\pi, r, \sum \lambda_i \pi_i)$, the transportation cost.
2. $est\_cost_{opt}(k, r) = \min_{x \in \mathcal{X}} (\omega \wedge r(x) - \sum \lambda_i \omega_i(x))$.

A knowledge state algorithm services a request sequence $\varrho = r^1, r^2, \ldots, r^n$ by picking a sequence of knowledge states, $k^0 k^1, \ldots, k^n$, as follows.

1. k^0 is the cone knowledge state over x^0.
2. For each $1 \leq t \leq n$, let $\mathcal{A}(k^{t-1}, r^t) = (k_1, \ldots, k_m, \lambda_1, \ldots, \lambda_m)$. Using randomization, $\mathcal{A}$ chooses $k^t \leftarrow k_i$ for some i, where each k_i is chosen with probability λ_i.

We define

$$cost^t_{\mathcal{A}}(\varrho) = cost_{\mathcal{A}}\left(k^{t-1}, r^t\right),$$

$$est\_cost^t_{opt}(\varrho) = est\_cost_{opt}\left(k^{t-1}, r^t\right),$$

$$cost_{\mathcal{A}}(\varrho) = \sum_{t=1}^{n} cost^t_{\mathcal{A}}(\varrho),$$

$$est\_cost_{opt}(\varrho) = \sum_{t=1}^{n} est\_cost^t_{opt}(\varrho).$$

Note that the above costs are random variables. Note also that the definition of $cost_{\mathcal{A}}$ agrees with the definition of the cost of a mixed model algorithm given in Sect. 2.

We say that $\mathcal{A}$ *is C-ks-competitive* if there is a constant K such that

$$Exp(cost_{\mathcal{A}}(\varrho)) \leq C \cdot Exp\big(est\_cost_{opt}(\varrho)\big) + K$$

for any request sequence ϱ.

Lemma 5 *Given a request sequence $\varrho = r^1, \ldots, r^n$, let $k^t = (\pi^t, \omega^t)$ be the service computed by $\mathcal{A}$. Then, $Exp(est\_cost_{opt}(\varrho) + \omega^n(x)) \leq cost^{\varrho}_{opt}(x)$ for all $x \in \mathcal{X}$.*

Proof Let $x^0, x^1, \ldots, x^n = x \in \mathcal{X}$ be the optimal service of ϱ that ends in x.

By definition of optimal cost, $\sum_{t=1}^{n} cost(x^{t-1}, r^t, x^t) = cost_{opt}(x)$.

By definition of estimated optimal cost, $Exp(est\_cost^t_{opt}(\varrho) + \omega^t(x^t)) \leq Exp((\omega^{t-1} \wedge r^t)(x^t))$ for all t.

By definition of update, $Exp((\omega^{t-1} \wedge r^t)(x^t)) \leq Exp(\omega^{t-1}(x^{t-1})) + cost(x^{t-1}, r^t, x^t)$ for all t.

Recall that $\omega^0(x^0) = 0$. Combining the above, we have

$$\begin{aligned} Exp\big(est\_cost_{opt}(\varrho) + \omega^n(x)\big) &= \sum_{1\le t\le n} Exp\Big(est\_cost^t_{opt}(\varrho) + \omega^t(x^t) - \omega^{t-1}(x^{t-1})\Big) \\ &\le \sum_{1\le t\le n} Exp\Big(\big(\omega^{t-1} \wedge r^t\big)(x^t) - \omega^{t-1}(x^{t-1})\Big) \\ &\le \sum_{1\le t\le n} \Big(cost(x^{t-1}, r^t, x^t)\Big) \\ &= cost^{\varrho}_{opt}(x) \end{aligned}$$

and we are done. □

Lemma 6 *If a knowledge state algorithm $\mathcal{A}$ is C-ks-competitive, then $\mathcal{A}$ is C-competitive.*

Proof Let K be the constant given in the definition of C-ks-competitiveness for any request sequence ϱ, $Exp(cost_{\mathcal{A}}(\varrho)) \le C \cdot Exp(est\_cost_{opt}(\varrho)) + K \le C \cdot cost_{opt}(\varrho) + K$, by Lemma 5. □

We now define a *ks-potential*, for a given knowledge state algorithm $\mathcal{A}$. Let Φ be a real-valued function on knowledge states. Let $C \ge 1$. Then we say that Φ is a *C-ks-potential for $\mathcal{A}$* if

1. $\Phi(k) \ge 0$ for any k.
2. If $k = (\pi, \omega)$ is the current knowledge state and r is the next request, let $\{k_i = (\pi_i, \omega_i)\}$ be the subsequents of that request, and $\{\lambda_i\}$ the weights of the subsequents. Let $\Delta\Phi(k, r) = \sum_{i=1}^m \lambda_i \Phi(\pi_i, \omega_i) - \Phi(\pi, \omega)$. Then

$$cost_{\mathcal{A}}(k, r) + \Delta\Phi(k, r) \le C \cdot est\_cost_{opt}(k, r).$$

Lemma 7 *If a knowledge state algorithm $\mathcal{A}$ has a C-ks-potential, then $\mathcal{A}$ is C-competitive.*

Proof The proof follows easily from the definition of a C-ks-potential and Lemmas 5 and 6 by straightforward arguments. Let $\varrho = r^1, \dots, r^n$ be a request sequence. Let $k^1, \dots, k^n$ be the sequence of knowledge states of $\mathcal{A}$ given the input ϱ, where $k^t = (\pi^t, \omega^t)$. Let $\Phi^t = \Phi(k^t)$, a random variable for each t. Note that Φ^0 is a constant. Let $\Delta^t\Phi = \Delta\Phi(k^{t-1}, r^t)$. Note that $Exp(\Delta^t\Phi) = Exp(\Phi^t - \Phi^{t-1})$. Let $x \in \mathcal{X}$ be the configuration of the optimal algorithm after n steps. Then

$$\begin{aligned} &C \cdot cost_{opt}(\varrho) - Exp(cost_{\mathcal{A}}(\varrho)) \\ &\quad \ge C \cdot Exp\big(\omega^n(x) + est\_cost_{opt}(\varrho)\big) - Exp(cost_{\mathcal{A}}(\varrho)) \\ &\quad = C \cdot Exp\left(\omega^n(x) + \sum_{t=1}^n est\_cost^t_{opt}(\varrho)\right) - Exp\left(\sum_{t=1}^n cost^t_{\mathcal{A}}(\varrho)\right) \end{aligned}$$

$$= Exp\left(C \cdot \omega^n(x) + \sum_{t=1}^{n}\left(C \cdot est\_cost^t_{opt}(\varrho) - cost^t_{\mathcal{A}}(\varrho)\right)\right)$$

$$= Exp\left(C \cdot \omega^n(x) + \Phi^n + \sum_{t=1}^{n}\left(C \cdot est\_cost^t_{opt}(\varrho) - cost^t_{\mathcal{A}}(\varrho) - \Delta^t\Phi\right)\right) - \Phi^0$$

$$\geq Exp\big(C \cdot \omega^n(x) + \Phi^n\big) - \Phi^0 \geq -\Phi^0.$$

The first inequality above is from Lemma 5. The last two inequalities are from the definition of a C-ks-potential. It follows that $Exp(cost_{\mathcal{A}}(\varrho)) \leq C \cdot cost_{opt}(\varrho) + \Phi^0$, and, by Lemma 6, we are done. □

4 Knowledge State Algorithms for the Paging Problem

We now consider the k-paging problem for fixed $k \geq 2$, which we formally define below.

1. There is a set of *pages*, $\mathcal{Q}$.
2. We define a *k-set* to be a set of exactly k pages. Let $\mathcal{X}$ be the set of all k-sets. If the configuration of an algorithm is $X \in \mathcal{X}$, that means that the pages that constitute X are in the cache.
3. The initial configuration is the initial cache, which we call X^0.
4. If $X, Y \in \mathcal{X}$, then $d(X, Y) = \|X, Y\|$ is the cost of changing the cache from X to Y, the cardinality of $X - Y$, since we assume that it costs 1 to eject a page and bring in a new page.
5. A task is simply the request that a particular page move to the cache, and thus $\mathcal{R} = \mathcal{Q}$, the set of all pages. If a page r is requested, the algorithm must ensure that r is in the cache at some point as it moves between configurations; for example, the algorithm could move r into the cache, and then move it back out.[1] Thus, for any $X, Y \in \mathcal{X}$ and any $r \in \mathcal{Q}$, we have

$$cost(X, r, Y) = \begin{cases} 2 & \text{if } X = Y,\ r \notin X, \\ \|X, Y\| & \text{if } r \in X \text{ or } r \in Y, \\ \|X, Y\| + 1 & \text{otherwise.} \end{cases}$$

Without loss of generality, there are never two consecutive requests to the same page.

Bar Notation for the Paging Problem Koutsoupias and Papadimitriou completely characterize reachable offset functions for the k-paging problem [14, 15].

Lemma 8 *Every offset function ω is supported by a set of configurations on which ω is zero. Furthermore, there exists an ordered k-tuple of sets of pages $(L_1, L_2, \ldots, L_k)$ called the* sequence of layers *of ω such that*

[1] This seems unnecessary, and in fact, without loss of generality, no algorithm will move a page out of the cache after it is requested. But the fact that this is move is permitted is necessary for our analysis.

1. $|L_1| = 1$. (Without loss of generality, $L_1 = \{r\}$, where r is the last request point.)
2. $L_i \cap L_j$ *for all* $1 \leq i, j \leq k$ *with* $i \neq j$.
3. *If* $X \in \mathcal{X}$, *then* $X \in supp(\omega)$ *if and only if* $|X \cap \bigcup_{i \leq j} L_i| \geq j$ *for all* $1 \leq j \leq k$.

We call L_i the ith *layer* of ω. Let $(L_1, L_2, \ldots, L_k)$ be the sequence of layers of a reachable offset function ω. If $x \in \bigcup_{1 \leq i \leq k} L_i$, we say that x is a *supporting* page of ω; otherwise, we say that x is an *external* page.

We define the *layer type* of ω to be the sequence $(|L_1|, |L_2|, \ldots, |L_k|)$. The sequence of layers of ω need not be unique, but the layer type is.

The *bar notation* for ω is the string of symbols $L_1 \,|\, L_2 \,|\cdots L_k \,|$. We allow each set to be written without commas or braces. For example, if $k = 3$, $L_1 = \{a\}$, $L_2 = \{b\}$, and $L_3 = \{c, d\}$, we write $\omega = a \,|\, b \,|\, c\,d \,|$. In this case, the layer type of ω is $(1, 1, 2)$.

If $\omega = L_1 \,|\, L_2 \,|\cdots L_k \,|$ and page r is requested, we can compute the bar notation for $\omega \wedge r$.

$$\omega \wedge r = \begin{cases} r \,|\, L_2 \,|\cdots|\, L_{j-1} \,|\, L_j \cup L_{j+1} - \{r\} \,|\, L_{j+2} \,|\cdots|\, L_k \,| & \text{if } r \in L_j \text{ for } j < k, \\ r \,|\, L_2 \,|\cdots|\, L_{k-1} \,| & \text{if } r \in L_k, \\ r \,|\, L_1 \cup L_2 \,|\, L_3 \,|\cdots|\, L_k \,| + 1 & \text{otherwise, } \textit{i.e.}\text{, if } r \text{ is external.} \end{cases}$$

For example, if a, b, c, d, e, f, and g are distinct pages, then

$$a \,|\, b\,c \,|\, d\,e \,|\, f\,g \,| \wedge d = d \,|\, a \,|\, b\,c \,|\, e\,f\,g \,|,$$

$$a \,|\, b\,c \,|\, d\,e \,| \wedge d = d \,|\, a \,|\, b\,c \,|,$$

$$a \,|\, b \,|\, c\,d \,| \wedge e = e \,|\, a\,b \,|\, c\,d \,| + 1.$$

We refer the reader to the rules of updating given above, and to [1, 15], to verify the above equations.

Given an offset function ω, two supporting pages a and b are *equivalent* if $supp(\omega)$ is invariant under exchange of a and b. We say that (k, r) is a *lazy* move if r is a supporting page of k, and that r is a *lazy request*. Note that if r is a lazy request, then $\omega \wedge r = \omega \,\overline{\wedge}\, r$ is also an offset function; in fact, $supp(\omega \wedge r) \subseteq supp(\omega)$ in that case. But if r is an external page, then the minimum value of $\omega \wedge r$ is 1, since the new page must be brought into the cache at a cost of 1. In that case, $\omega \wedge r = \omega \,\overline{\wedge}\, r + 1$.

Bookmarks for the k-Paging Problem Suppose that $\mathcal{A}$ is a knowledge state algorithm for the k-paging problem. At any given step, the current knowledge state will be of the form (π, ω), where ω is an offset function. Suppose that the layer sequence of ω is $(L_1, L_2, \ldots, L_k)$, *i.e.*, ω has layer type $(|L_1|, |L_2|, \ldots, |L_k|)$. Let $S = S(\omega) = \bigcup L_i$, the set of supporting pages of ω, and suppose that $|S(\omega)| = m$. We can assume that π is supported by S. An actual implementation of $\mathcal{A}$ will be a behavioral algorithm $\mathcal{B}$, as described in the proof of Lemma 1. We can assume that the probability that $\mathcal{B}$ contains any external page is zero. At any given step, $\mathcal{B}$ has k pages in its cache, but it must remember the function ω, which means it must keep track of the $m - k$ pages that are known but not in the cache. We will say that $\mathcal{B}$'s memory must contain $m - k$ *bookmarks*, one for each known page which is not in its cache.

Initially, the support of the offset function is simply the set of pages in the cache. But, after n requests, the support of the offset function ω could grow to as many as $k+n$ pages, requiring n bookmarks, if $\mathcal{B}$ needs to remember ω entirely. It is quite impractical to implement a paging algorithm whose memory requirements are unbounded. Fortunately, Achlioptas *et al.* [1] prove that the optimal competitiveness of H_k can be achieved with finite memory. Their algorithm, EQUITABLE, which we describe in Sect. 4.3, uses $O(k^2 \log k)$ bookmarks regardless of the length of the request sequence.

In this paper, we improve on that result. We first present an algorithm for the 2-paging problem that requires one bookmark, and an algorithm for the 3-paging problem that requires two bookmarks. In Sect. 4.3, we define an algorithm for the k-paging problem, for general k, that requires $2k$ bookmarks.

4.1 An Optimally Competitive Knowledge State Algorithm for the 2-Paging Problem with One Bookmark

We now give a knowledge state algorithm, K2, for the 2-paging problem, which uses only one bookmark, and whose competitiveness is $H_2 = \frac{3}{2}$.

1. The set of knowledge states of K2 is $\mathcal{K} = \{A^{a,b}\} \cup \{B^{a,b,c}\}$, where
   (a) $A^{a,b} = (\{a,b\},\ a\,|\,b\,|)$ for any distinct pages a, b. Note that $A^{a,b}$ is the cone knowledge state over $\{a,b\}$.
   (b) $B^{a,b,c} = (\frac{1}{2}\{a,b\} + \frac{1}{2}\{a,c\},\ a\,|\,b\,c\,|)$ for any distinct pages a, b, c.
   There can be more than one name for the same knowledge state, since the bar notation of an offset function is not unique. In particular, $A^{a,b} = A^{b,a}$, and $B^{a,b,c} = B^{a,c,b}$.
2. For any distinct pages a, b, c, r:
   (a) If the knowledge state is $A^{a,b}$, a request to either a or b is *trivial*. If the knowledge state is $B^{a,b,c}$, a request to a is trivial. K2 services trivial requests by doing nothing. Without loss of generality, there are never any trivial requests.
   (b) $K2(A^{a,b}, r) = (B^{r,a,b}, 1)$. This move is illustrated in Fig. 1.
   (c) $K2(B^{a,b,c}, b) = (A^{b,a}, 1)$. This move is illustrated in Fig. 2.
   (d) $K2(B^{a,b,c}, r) = (B^{r,a,b}, B^{r,a,c}, B^{r,b,c}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. This move is illustrated in Fig. 3.

We now prove the $\frac{3}{2}$-competitiveness of K2. Define

1. $\Phi(A^{a,b}) = 0$.
2. $\Phi(B^{a,b,c}) = \frac{1}{2}$.

Lemma 9 *Φ is a $\frac{3}{2}$-ks-potential for K2.*

Proof We need to prove that each move of K2 satisfies the inequalities given in the definition of ks-competitiveness in Sect. 3. Trivially, $\Phi(k) \geq 0$. We need to verify that

$$cost_{K2}(k,r) + \Delta\Phi(k,r) \leq C \cdot est\_cost_{opt}(k,r)$$

holds for each of the three non-trivial moves.

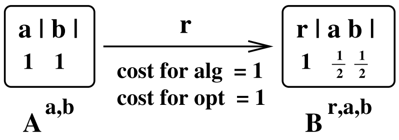

Fig. 1 Move $(A^{a,b}, r)$

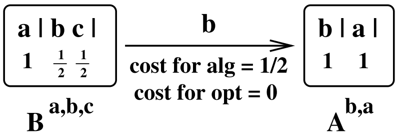

Fig. 2 Move $(B^{a,b,c}, b)$

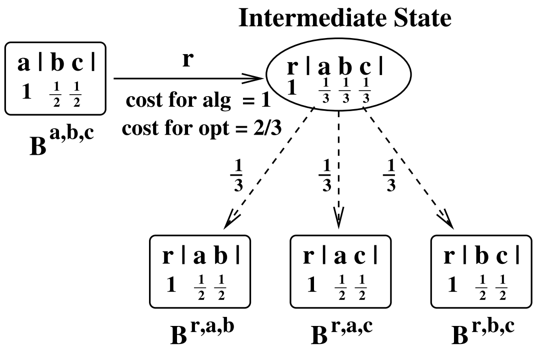

Fig. 3 Move $(B^{a,b,c}, r)$

Case I: $(k, r) = (A^{a,b}, r)$, where $r \notin \{a, b\}$.

Since K2 must eject a page, $cost_{K2}(k, r) = 1$. Since r is an external page, $\omega \wedge r = r\,|\,a\,b\,| + 1$, and thus $est\_cost_{opt}(k, r) = 1$. Thus

$$cost_{K2}(A^{a,b}, r) + \Phi(B^{r,a,b}) = \frac{3}{2} = \Phi(A^{a,b}) + \frac{3}{2} est\_cost_{opt}(A^{a,b}, r)$$

and we are done.

Case II: $(k, r) = (B^{a,b,c}, b)$.

The probability is $\frac{1}{2}$ that the cache contains b, and $\frac{1}{2}$ that it does not; thus $cost_{K2}(B^{a,b,c}, b) = \frac{1}{2}$. Since $B^{a,b,c} \wedge b = A^{b,a}$, we have $est\_cost_{opt} = 0$. Thus

$$cost_{K2}(B^{a,b,c}, b) + \Phi(A^{b,a}) = \frac{1}{2} = \Phi(B^{a,b,c}) + \frac{3}{2} est\_cost_{opt}(B^{a,b,c}, b)$$

and we are done.

Case III: $(k, r) = (B^{a,b,c}, r)$, where $r \notin \{a, b, c\}$.

Since K2 must eject a page, $cost_{K2}(k, r) = 1$. To compute $est\_cost_{opt}(B^{a,b,c}, r)$, we compute $\omega \wedge r$ and $\bar{\omega} = \sum_i \omega_i$. $\omega \wedge r = r\,|a\,b\,c\,| + 1$. It suffices to compute that function on $supp(\bar{\omega}) = \{\{r, a\}, \{r, b\}, \{r, c\}\}$. Since $\omega_1(\{r, a\}) = 0$, $\omega_2(\{r, a\}) = 0$, and $\omega_3(\{r, a\}) = 1$, we have $\bar{\omega}(\{r, a\}) = \frac{1}{3}$. Similarly, $\bar{\omega}(\{r, b\}) = \bar{\omega}(\{r, c\}) = \frac{1}{3}$. Thus,

$est\_cost(B^{a,b,c}, r) = \frac{2}{3}$. Thus

$$cost_{K2}(B^{a,b,c}, r) + \Delta\Phi = 1 = \frac{3}{2} est\_cost_{opt}(B^{a,b,c}, r)$$

and we are done. □

Theorem 3 K2 *is* $\frac{3}{2}$-*competitive.*

Proof From Lemmas 6, 7, and 9. □

In Figs. 1, 2, and 3, each knowledge state is shown as a rectangle containing the offset function written in bar notation. The probability that a given page is in the cache is indicated by a number under the name of the page.

We note that the number of known pages, *i.e.*, pages contained in a support configuration, is never more than three. The number three is minimal, as given by the theorem below:

Theorem 4 *There is no knowledge state algorithm for the* 2-*paging problem that is* $\frac{3}{2}$-*competitive as a knowledge state algorithm, and which never has more than two known pages*, i.e., *no bookmarks.*

Proof If a knowledge state algorithm for the 2-paging problem never has more than two known pages, then it can have no bookmarks, hence is trackless. By Theorem 2 of [4], there is no $\frac{3}{2}$-competitive trackless online algorithm for the 2-paging problem. □

4.2 An Optimally Competitive Knowledge State Algorithm for the 3-Paging Problem with Two Bookmarks

We now give a knowledge state algorithm, K3, for the 3-paging problem, which uses only two bookmarks, and whose competitiveness is $H_3 = \frac{11}{6}$.

1. The set of knowledge states of K3 is $\mathcal{K} = \{A^{a,b,c}\} \cup \{C^{a,b,c,d}\} \cup \{D^{a,b,c,d,e}\} \cup \{E^{a,b,c,d,e}\} \cup \{F^{a,b,c,d,e}\}$, where
   (a) $A^{a,b,c} = (\{a,b,c\}, a\,|\,b\,|\,c\,|)$ for any distinct pages a, b, c. Note that $A^{a,b,c}$ is the cone knowledge state over $\{a,b,c\}$.
   (b) $B^{a,b,c,d} = (\frac{1}{3}\{a,b,c\} + \frac{1}{3}\{a,b,d\} + \frac{1}{3}\{a,c,d\}, a\,|\,b\,c\,|\,d\,|)$ for any distinct pages a, b, c, d.
   (c) $C^{a,b,c,d} = (\frac{1}{2}\{a,b,c\} + \frac{1}{2}\{a,b,d\}, a\,|\,b\,|\,c\,d\,|)$ for any distinct pages a, b, c, d.
   (d) $D^{a,b,c,d,e} = (\frac{1}{6}\{a,b,c\} + \frac{1}{6}\{a,b,d\} + \frac{1}{6}\{a,b,e\} + \frac{1}{6}\{a,c,d\} + \frac{1}{6}\{a,c,e\} + \frac{1}{6}\{a,d,e\}, a\,|\,b\,c\,d\,|\,e\,|)$ for any distinct pages a, b, c, d, e.
   (e) $E^{a,b,c,d,e} = (\frac{1}{3}\{a,b,c\} + \frac{1}{3}\{a,b,d\} + \frac{1}{3}\{a,b,e\}, a\,|\,b\,|\,c\,d\,e\,|)$ for any distinct pages a, b, c, d, e.
   (f) $F^{a,b,c,d,e} = (\frac{1}{3}\{a,b,c\} + \frac{1}{6}\{a,b,d\} + \frac{1}{6}\{a,b,e\} + \frac{1}{6}\{a,c,e\} + \frac{1}{6}\{a,d,e\}, a\,|\,b\,c\,|\,d\,e\,|)$ for any distinct pages a, b, c, d, e.
2. Moves of K3 are as follows. In the list below, we assume that distinct letters represent distinct pages.

Table 1 Moves with one subsequent

Move	Subsequent
$(A^{a,b,c}, r)$	$B^{r,a,b,c}$
$(B^{a,b,c,d}, b)$	$C^{b,a,c,d}$
$(B^{a,b,c,d}, c)$	$C^{c,a,b,d}$
$(B^{a,b,c,d}, d)$	$C^{d,a,b,c}$
$(B^{a,b,c,d}, r)$	$D^{r,a,b,c,d}$
$(C^{a,b,c,d}, c)$	$A^{c,a,b}$
$(C^{a,b,c,d}, d)$	$A^{d,a,b}$
$(C^{a,b,c,d}, r)$	$F^{r,a,b,c,d}$
$(D^{a,b,c,d,e}, b)$	$E^{b,a,c,d,e}$
$(D^{a,b,c,d,e}, c)$	$E^{c,a,b,d,e}$
$(D^{a,b,c,d,e}, d)$	$E^{d,a,b,c,e}$
$(D^{a,b,c,d,e}, e)$	$E^{e,a,b,c,d}$
$(E^{a,b,c,d,e}, c)$	$A^{c,a,b}$
$(E^{a,b,c,d,e}, d)$	$A^{d,a,b}$
$(E^{a,b,c,d,e}, e)$	$A^{e,a,b}$
$(F^{a,b,c,d,e}, b)$	$E^{b,a,c,d,e}$
$(F^{a,b,c,d,e}, c)$	$E^{c,a,b,d,e}$
$(F^{a,b,c,d,e}, d)$	$C^{d,a,b,c}$
$(F^{a,b,c,d,e}, e)$	$C^{e,a,b,c}$

(a) Trivial moves are $(A^{a,b,c}, a)$, $(A^{a,b,c}, b)$, $(A^{a,b,c}, c)$, $(B^{a,b,c,d}, a)$, $(C^{a,b,c,d}, a)$, $(C^{a,b,c,d}, b)$, $(D^{a,b,c,d,e}, a)$, $(E^{a,b,c,d,e}, a)$, $(E^{a,b,c,d,e}, b)$, and $(F^{a,b,c,d,e}, a)$. Without loss of generality, there are never any trivial requests.

(b) Any move where the requested page is known, or where there are fewer than five known pages, has just one subsequent. The table lists (see Table 1) all such moves. Some moves are equivalent; equivalences classes are separated by empty lines.

(c) There are three cases where there are five pages and the request is to an external page.

i. $K3(D^{a,b,c,d,e}, r) = (D^{r,a,b,c,d}, D^{r,a,b,c,e}, D^{r,a,b,d,e}, D^{r,a,c,d,e}, D^{r,b,c,d,e}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$.

ii. $K3(E^{a,b,c,d,e}, r) = (A^{r,a,b}, 1)$.

iii. $K3(F^{a,b,c,d,e}, r) = (B^{r,a,b,c}, F^{r,a,b,d,e}, F^{r,a,c,d,e}, F^{r,b,c,d,e}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$.

We now prove the $\frac{11}{6}$-competitiveness of K3. Define

1. $\Phi(A^{a,b,c}) = 0$.
2. $\Phi(B^{a,b,c,d}) = \frac{5}{6}$.
3. $\Phi(C^{a,b,c,d}) = \frac{1}{2}$.
4. $\Phi(D^{a,b,c,d,e}) = \frac{5}{3}$.
5. $\Phi(E^{a,b,c,d,e}) = 1$.
6. $\Phi(F^{a,b,c,d,e}) = \frac{4}{3}$.

Lemma 10 *Φ is an $\frac{11}{6}$-ks-potential for* K3.

Proof We need to prove that each move of K3 satisfies the inequalities given in the definition of ks-competitiveness in Sect. 3. Trivially, $\Phi(k) \geq 0$. We need to verify that

$$cost_{K3}(k,r) + \Delta\Phi(k,r) \leq \frac{11}{6} \cdot est\_cost_{opt}(k,r)$$

holds for each of the twelve equivalence classes of non-trivial moves.

Case I: $(k, r) = (B^{a,b,c,d}, b)$. Since this is a lazy request, $est\_cost_{opt} = 0$. $cost_{K3} = \frac{1}{3}$, the probability that b is not in the cache. Thus

$$cost_{K3}(B^{a,b,c,d}, b) + \Phi(C^{b,a,c,d}) = \frac{5}{6} = \Phi(B^{a,b,c,d}).$$

Case II: $(k, r) = (C^{a,b,c,d}, c)$. Since this is a lazy request, $est\_cost_{opt} = 0$. $cost_{K3} = \frac{1}{2}$, the probability that c is not in the cache. Thus

$$cost_{K3}(C^{a,b,c,d}, c) + \Phi(A^{c,a,b}) = \frac{1}{2} = \Phi(C^{a,b,c,d}).$$

Case III: $(k, r) = (D^{a,b,c,d,e}, b)$. Since this is a lazy request, $est\_cost_{opt} = 0$. $cost_{K3} = \frac{1}{2}$, the probability that b is not in the cache. Thus

$$cost_{K3}(D^{a,b,c,d,e}, b) + \Phi(E^{b,a,c,d,e}) = \frac{3}{2} < \frac{5}{3} = \Phi(D^{a,b,c,d,e}).$$

Case IV: $(k, r) = (E^{a,b,c,d,e}, c)$. Since this is a lazy request, $est\_cost_{opt} = 0$. $cost_{K3} = \frac{2}{3}$, the probability that c is not in the cache. Thus

$$cost_{K3}(E^{a,b,c,d,e}, c) + \Phi(A^{c,a,b}) = \frac{2}{3} < 1 = \Phi(D^{a,b,c,d,e}).$$

Case V: $(k, r) = (F^{a,b,c,d,e}, b)$. Since this is a lazy request, $est\_cost_{opt} = 0$. $cost_{K3} = \frac{1}{3}$, the probability that b is not in the cache. Thus

$$cost_{K3}(F^{a,b,c,d,e}, b) + \Phi(E^{b,a,c,d,e}) = \frac{4}{3} = \Phi(F^{a,b,c,d,e}).$$

Case VI: $(k, r) = (F^{a,b,c,d,e}, d)$. Since this is a lazy request, $est\_cost_{opt} = 0$. $cost_{K3} = \frac{2}{3}$, the probability that d is not in the cache. Thus

$$cost_{K3}(F^{a,b,c,d,e}, d) + \Phi(C^{d,a,b,c}) = \frac{7}{6} < \frac{4}{3} = \Phi(F^{a,b,c,d,e}).$$

Case VII: $(k, r) = (A^{a,b,c}, r)$, where $r \notin \{a, b, c\}$.

Since K3 must eject a page, $cost_{K3}(k, r) = 1$. Since r is an external page, $\omega \wedge r = r\,|\,a\,b\,c\,| + 1 = B^{r,a,b,c} + 1$, and thus $est\_cost_{opt}(k, r) = 1$. Thus

$$cost_{K3}(A^{a,b,c}, r) + \Phi(B^{r,a,b,c}) = \frac{11}{6} = \Phi(A^{a,b,c}) + \frac{11}{6} est\_cost_{opt}(A^{a,b,c}, r).$$

Case VIII: $(k, r) = (B^{a,b,c,d}, r)$, where $r \notin \{a, b, c, d\}$. Since K3 must eject a page, $cost_{K3}(k, r) = 1$. Since r is an external page, $\omega \wedge r = r\,|\,a\,b\,c\,d\,| + 1 = D^{r,a,b,c,d} + 1$, and thus $est\_cost_{opt}(k, r) = 1$. Thus

$$cost_{K3}(B^{a,b,c,d}, r) + \Phi(D^{r,a,b,c,d}) = \frac{8}{3} = \Phi(B^{a,b,c,d}) + \frac{11}{6} est\_cost_{opt}(B^{a,b,c,d}, r).$$

Case IX: $(k, r) = (C^{a,b,c,d}, r)$, where $r \notin \{a, b, c, d\}$. Since K3 must eject a page, $cost_{K3}(k, r) = 1$. Since r is an external page, $\omega \wedge r = r \,|\, a\, b \,|\, c\, d \,|\, + 1 = F^{r,a,b,c,d} + 1$, and thus $est\_cost_{opt}(k, r) = 1$. Thus

$$cost_{K3}(C^{a,b,c,d}, r) + \Phi(F^{r,a,b,c,d}) = \frac{7}{3} = \Phi(C^{a,b,c,d}) + \frac{11}{6} est\_cost_{opt}(C^{a,b,c,d}, r).$$

Case X: $(k, r) = (D^{a,b,c,d,e}, r)$, where $r \notin \{a, b, c, d, e\}$. Since K3 must eject a page, $cost_{K3}(k, r) = 1$. Since r is an external page, $\omega \wedge r = r \,|\, a\, b\, c\, d\, e \,|\, + 1$. Averaging the five subsequents, we obtain $\bar{\omega}(X) = \frac{2}{5}$ for each $X \in supp(\omega \wedge r)$, hence $est\_cost_{opt}(k, r) = \frac{3}{5}$. Since all subsequents have the same potential as the start configuration, $\Delta\Phi = 0$. Thus

$$cost_{K3}(D^{a,b,c,d,e}, r) + \Delta\Phi = 1 < \frac{11}{10} = \frac{11}{6} est\_cost_{opt}(D^{a,b,c,d,e}, r).$$

Case XI: $(k, r) = (E^{a,b,c,d,e}, r)$, where $r \notin \{a, b, c, d, e\}$. Since K3 must eject a page, $cost_{K3}(k, r) = 1$. Since r is an external page, $\omega \wedge r = r \,|\, a\, b \,|\, c\, d\, e \,|\, + 1$, while $\bar{\omega} = r \,|\, a \,|\, b \,|$. Checking each $X \in supp(\omega \wedge r)$, we verify that $est\_cost_{opt}(k, r) = 0$. Thus

$$cost_{K3}(E^{a,b,c,d,e}, r) + \Phi(A^{r,a,b}) = 1 = \Phi(E^{a,b,c,d,e}).$$

Case XII: $(k, r) = (F^{a,b,c,d,e}, r)$, where $r \notin \{a, b, c, d, e\}$. Since K3 must eject a page, $cost_{K3}(k, r) = 1$. Since r is an external page, $\omega \wedge r = r \,|\, a\, b\, c \,|\, d\, e \,|\, + 1$. Averaging over the four subsequents and checking all $X \in supp(\omega \wedge r)$, we find that the minimum value of $(\omega \wedge r)(X) - \bar{\omega}(X)$ is $\frac{1}{2}$. Averaging the potentials of the subsequents, we find that $Exp(\Delta\Phi) = -\frac{1}{6}$. Thus

$$cost_{K3}(F^{a,b,c,d,e}, r) + Exp(\Delta\Phi) = \frac{5}{6} < \frac{11}{12} = \frac{11}{6} est\_cost_{opt}(F^{a,b,c,d,e}, r).$$

Thus, the inequality holds for every move, and we are done. □

Theorem 5 K3 *is* $\frac{11}{6}$*-competitive.*

Proof From Lemmas 6, 7, and 10. □

4.3 Knowledge State Algorithms for the k-Paging Problem for General k

We start by reviewing the algorithm EQUITABLE given by Achlioptas, Chrobak, and Noga [1].

The Algorithm EQUITABLE This algorithm is a randomized algorithm for the k-paging problem that is H_k-competitive and has space complexity $O(k^2 \log k)$. We briefly review EQUITABLE; the reader is referred to [1] for further details.

Fix k. For the offset function $\omega = L_1 \,|\, L_2 \,| \cdots L_k \,|$, let π^ω be the distribution on k-sets of pages given in [1]. That distribution, which we call the EQUITABLE distribution, and denote π^ω, can be described by defining a randomized algorithm for choosing the cache, X:

1. Initialize X to be the empty set.
2. Let $T = \bigcup L_i$, the set of known pages.
3. Execute the following loop until $|X| = k$:
   (a) Select $x \in T$ uniformly at random.
   (b) Delete x from T.
   (c) If $|(X \cup \{x\}) \cap L_i| \geq i + |X| + 1 - k$ for all $1 \leq i \leq k$ (*i.e.*, if $X \cup \{x\}$ is a subset of some member of $supp(\omega)$) then $X \leftarrow X \cup \{x\}$.

From [1] we have:

Lemma 11 $X \in supp(\omega) \iff \pi^{\omega}(X) > 0$.

EQUITABLE is defined as a knowledge state algorithm, as follows.

1. Let $M = \lceil 5k^2 H_k \rceil$. Let $\mathcal{K}$ be the set of all knowledge states of the form $k^{\omega} = (\pi^{\omega}, \omega)$, where ω is a reachable offset function for the k-paging problem which has at most M supporting pages.
2. We define EQUITABLE(k^{ω}, r) for any $k^{\omega} \in \mathcal{K}$ and any page r, as follows.
   (a) If r is a supporting page of ω, or if ω has fewer than M supporting pages, we let $k^{\omega \overline{\wedge} r} \in \mathcal{K}$ be the single subsequent of the move (k^{ω}, r).
   (b) Otherwise, *i.e.*, if ω has exactly M supporting pages and r is an external page, let the subsequents of the move (k^{ω}, r) be the set of cone knowledge states $\{\kappa^X : X \in supp(\omega \overline{\wedge} r)\}$. Let the weight of the subsequent κ^X be $\pi^{\omega \overline{\wedge} r}(X)$.

Theorem 6 EQUITABLE *requires* $O(k^2 \log k)$ *bookmarks.*

Proof There are never more than $M - k$ bookmarks. □

We refer the reader to [1] for the proof of the main result:

Theorem 7 EQUITABLE *is* H_k*-competitive.*

The New Algorithm K_EQUITABLE We will now describe an H_k-competitive (thus optimally competitive) algorithm for the k-paging problem, which keeps track of only $3k$ pages. We call the algorithm K_EQUITABLE. As described above, the algorithm EQUITABLE is a knowledge state algorithm, though it was not defined in these terms in [1]. The set of knowledge states of K_EQUITABLE is a proper subset of the set of knowledge states of EQUITABLE.

K_EQUITABLE is defined as a knowledge state algorithm, as follows.

1. Let $M = 3k$. Let $\mathcal{K}$ be the set of all knowledge states of the form $k^{\omega} = (\pi^{\omega}, \omega)$, where ω is a reachable offset function for the k-paging problem which has at most M supporting pages. (The distribution π^{ω} is the EQUITABLE distribution defined above.)
2. We define K_EQUITABLE(k^{ω}, r) for any $k^{\omega} \in \mathcal{K}$ and any page r, as follows.
   (a) If r is a supporting page of ω, or if ω has fewer than M supporting pages, we let $k^{\omega \overline{\wedge} r} \in \mathcal{K}$ be the single subsequent of the move (k^{ω}, r).

(b) Otherwise, *i.e.*, if ω has exactly M supporting pages and r is an external page, write $\omega = L_1|L_2|\cdots|L_{k-1}|L_k|$. The move (k^{ω}, r) has just one subsequent, which is $k^{\tilde{\omega}}$, where $\tilde{\omega} = r|L_1|L_2|\cdots|L_{k-1}|$.

For convenience we introduce the following notation: If $\omega = L_1|L_2|\cdots|L_{k-1}|L_k|$, we define $S_i = \bigcup_{1\le j\le i} L_j$. Thus, S_k is the set of all supporting pages of ω. Let $m_i = |S_i|$.

We define the function Ψ to be the cost EQUITABLE or K_EQUITABLE incurs on a sequence of lazy requests ending when $|S_k| = k$. This is well defined due to the following observations given in [1]: For an offset function ω and page x, define $p_x = p_x^{\omega}$ to be the probability that the page x is in the cache, given that the distribution is π^{ω}. Equivalently let $p_x = \sum_{\{X\in S^k | x\in X\}} \pi^{\omega}(X)$. Note that $p_x > 0 \iff x \in S_k$.

Observation 1 *If $x \in L_i$ and $y \in L_j$, where $i \le j$, then $p_x \ge p_y$.*

Observation 2 *For any offset function, all sequences of lazy requests ending when $|S_k| = k$ have the same cost.*

Define $\Gamma = \Gamma(\omega)$ as follows:

$$\Gamma = \sum_{i=2}^{k} \Big(\frac{m_i}{i} + H_{i-1} - H_{m_i-1} - 1\Big).$$

We define $\Phi = \Psi + \Gamma$. We will show that Φ is an H_k-ks-potential for K_EQUITABLE, *i.e.*, that

$$cost + \Delta\Psi + \Delta\Gamma \le H_k \cdot est\_cost_{opt}$$

for any given move, where $cost = cost_{K\_Equitable}$ for that move.

In the discussion below, unprimed variables denote the values before a given request. Primed variables are the values after that request. Recall that $m_i \ge i$.

Lemma 12 *On a lazy request $r \in L_j$, $cost + \Delta\Psi + \Delta\Gamma \le H_k \cdot est\_cost_{opt}$.*

Proof Since Ψ is the cost for EQUITABLE to serve a lazy sequence of requests ending in a cone, on a lazy request $cost + \Delta\Psi = 0$, by the definition of Ψ. Also, for a lazy request $est\_cost_{opt} = 0$. Thus, it suffices to show that $\Delta\Gamma < 0$. We have

$$\begin{aligned}
\Delta\Gamma &= \sum_{i=2}^{k} \left(\frac{m_i'}{i} - H_{m_i'-1} - \frac{m_i}{i} + H_{m_i-1}\right) \\
&= \sum_{i=2}^{j} \left(\frac{m_{i-1} + 1 - m_i}{i} - H_{m_{i-1}} + H_{m_i-1}\right) \\
&= \sum_{i=2}^{j} \left(\frac{m_{i-1} + 1 - m_i}{i} + \sum_{\ell=m_{i-1}+1}^{m_i-1} \frac{1}{\ell}\right)
\end{aligned}$$

$$
\begin{aligned}
&\le \sum_{i=2}^{j}\left(\frac{m_{i-1}+1-m_i}{i}+\sum_{\ell=m_{i-1}+1}^{m_i-1}\frac{1}{i}\right)\\
&=0.
\end{aligned}
$$

The inequality follows from the fact that $\ell \ge m_{i-1}+1 \ge i$ for each index ℓ in the summation. □

Lemma 13 *On an external request $r \notin S_k$, where $m_k < 3k$, then $\Delta\Psi \le \sum_{i=2}^{k}\frac{1}{m_i}$.*

Proof If $k=1$ then $\Psi = \Psi' = 0$ which shows that the lemma is true for $k=1$. Assume $k>1$ and that the lemma is true for $k-1$.

For every page $x \ne r$, $p_x \ge p'_x$, since r is the only page that gains mass. Since $p_r = 0$, $p'_r = 1$, $\sum_{x\in S_k}(p_x - p'_x) = 1$, and $|S_k| = m_k$, there must be an item $x \in S_k$ for which $p_x - p'_x \le 1/m_k$.

If $x \in L_1$, pick $y \in L_2$. We have $p_y \le p_x$, but $p'_y = p'_x$ since x and y are equivalent for ω'. Hence it follows $p_y - p'_y \le p_x - p'_x$. Thus, without loss of generality, $x \notin L_1$.

Let this page $x \in L_j$ be the first item in the lazy request sequence which defines Ψ and Ψ'. Define the following offset functions:

$$
\begin{aligned}
\omega &= L_1|L_2|\cdots|L_k|,\\
\omega' &= r|L_1\cup L_2|L_3|\cdots|L_k|,\\
\omega\wedge x &= x|L_1|L_2|\cdots|L_{j-1}|L_j\cup L_{j+1}-\{x\}|\cdots|L_k|,\\
\omega'\wedge x &= x|r|L_1\cup L_2|L_3|\cdots|L_{j-1}|L_j\cup L_{j+1}-\{x\}|\cdots|L_k|,\\
\omega_{dropx} &= L_1|L_2|\cdots|L_{j-1}|L_j\cup L_{j+1}|\cdots|L_k|,\\
\omega'_{dropx} &= r|L_1\cup L_2|L_3|\cdots|L_{j-1}|L_j\cup L_{j+1}|\cdots|L_k|.
\end{aligned}
$$

Now we notice that

$$
\begin{aligned}
\Delta\Psi &\le \frac{1}{m_k} + \Psi(\omega'\wedge x) - \Psi(\omega\wedge x)\\
&= \frac{1}{m_k} + \Psi(\omega'_{dropx}) - \Psi(\omega_{dropx})\\
&\le \frac{1}{m_k} + \sum_{i=2}^{k-1}\frac{1}{m_i}\\
&= \sum_{i=2}^{k}\frac{1}{m_i},
\end{aligned}
$$

where the third line follows from the inductive hypothesis. □

Lemma 14 *On an external request $r \notin S_k$, if $m_k < 3k$,*

$$
cost + \Delta\Psi + \Delta\Gamma \le H_k \cdot est\_cost_{opt}.
$$

Proof Since $r \notin S_k$, $m'_i = m_i + 1$. Given Lemma 13, it follows that

$$\begin{aligned} cost + \Delta\Psi + \Delta\Gamma &\le 1 + \sum_{i=2}^{k} \frac{1}{m_i} + \sum_{i=2}^{k} \left(\frac{m'_i}{i} - H_{m'_i - 1} - \frac{m_i}{i} + H_{m_i - 1} \right) \\ &= 1 + \sum_{i=2}^{k} \frac{1}{m_i} + \sum_{i=2}^{k} \left(\frac{m_i + 1}{i} - H_{m_i} - \frac{m_i}{i} + H_{m_i - 1} \right) \\ &= \sum_{i=1}^{k} \frac{1}{i} \\ &= H_k \cdot est\_cost_{opt}. \end{aligned}$$

□

Lemma 15 *On an external request $r \notin S_k$ when $m_k = 3k$,*

$$cost + \Delta\Psi + \Delta\Gamma \le 0.$$

Proof An alternative way to implement this step is to place r in L_k and then request r. We can easily compute $\Delta\Gamma$ during this step. However, it is easier to compute $cost + \Delta\Psi$ when r is added to L_k, and then on the request, separately. The cost of this move is then separated into two parts.

When r is placed into L_k, the distribution must be adjusted and the lazy potential changes. The transportation cost necessary to adjust the distribution is p'_r. Since the cost on all lazy sequences is the same, we can compute the change in potential by considering a sequence which begins with some page $x \in L_k$. From Observation 1, $cost + \Delta\Psi = p_x - p'_x + p'_r = p_x \le \frac{k}{m_k}$.

When r is requested, then $cost + \Delta\Psi = 0$ because Ψ is the lazy potential. So it suffices to show that $\frac{k}{m_k} + \Delta\Gamma$ is no more than 0. We have

$$\begin{aligned} cost + \Delta\Psi + \Delta\Gamma &\le \frac{k}{m_k} + \sum_{i=2}^{k} \left(\frac{m'_i}{i} - H_{m'_i - 1} - \frac{m_i}{i} + H_{m_i - 1} \right) \\ &= \frac{1}{3} + \sum_{i=2}^{k} \left(\frac{m_{i-1} + 1 - m_i}{i} - H_{m_{i-1}} + H_{m_i - 1} \right) \\ &\le \frac{1}{3} + \left(\frac{k - m_k}{k} - H_{k-1} + H_{m_k - 1} \right) \\ &= -\frac{5}{3} + H_{3k-1} - H_{k-1} \\ &\le 0. \end{aligned}$$

To see the second inequality, first consider the case that $m_i = i$ for all $2 \le i \le k - 1$, the minimum possible values. In this case, we have equality. We can now verify that if we increase the value of any m_i for $2 \le i \le k - 1$, the difference between the formula on the third line and the formula on the second line does not decrease. □

Theorem 8 K_EQUITABLE *is an* H_k*-competitive*, $O(k)$ *memory*, *randomized algorithm for the* k*-paging problem.*

Proof The number of supporting pages is never more than $3k$, and thus the memory is $O(k)$. Lemmas 12, 14, and 15 show that $cost + \Delta\Psi + \Delta\Gamma \le H_k \cdot est\_cost_{opt}$ for every move. Note that $\Psi + \Gamma$ is initially 0 and never negative. When we sum over every request, we show that $cost_{K\_Equitable}(\varrho) \le H_k \cdot cost_{opt}(\varrho)$ for any request sequence ϱ. □

5 Summary of Further Results

One of the most challenging problems in online algorithms is to determine the exact randomized competitiveness of the k-server problem, that is, the minimum competitiveness of any randomized online algorithm for the server problem. Even in the case $k = 2$ it is not known whether its competitiveness is lower than 2, the known value of the deterministic competitiveness. This is surprising, since it seems intuitive that randomization should help. For the randomized 2-server problem we note that for the special case of the line, Bartal *et al.* [2] have given a randomized algorithm with a competitive ratio of better than 2. Unfortunately, the approach is specific to the line, using methods which do not appear to generalize to all metric spaces. Recently in [8], Bein *et al.* have designed a knowledge state algorithm with a competitive ratio of $\frac{19}{12}$ over Cross Polytope Spaces, and proved it is optimal against the oblivious adversary. Cross Polytope Spaces have been studied extensively starting as early as the 19th century; see Schläfli [16], as well as Fig. 4. They are part of a larger category $\mathcal{M}_{2,4}$, consisting of all metric spaces such that

- all distances are 1 or 2,
- $d(x, y) + d(y, z) + d(z, x) \le 4$.

We note that paging can be modeled as a server problem in uniform spaces. Thus $\mathcal{M}_{2,4}$ is "one step up" from uniform spaces and further work would focus on $\mathcal{M}_{3,6}$ and so forth.

Another result motivated by an application involved multiprocessor caching systems [7]. Multiprocessor systems with a global shared memory provide logically uniform data access. To hide latencies when accessing global memory, each processor makes use of a private cache. Several copies of a data item may exist concurrently in the system. To guarantee consistency when updating an item a processor must invalidate copies of the item in other private caches. To exclude the effect of classical paging faults, one assumes that each processor knows its own data access sequence, but does not know the sequence of future invalidations requested by other processors. Performance of a processor with this restriction can be measured against the optimal behavior of a theoretical omniscient processor, using competitive analysis. In [7], Bein *et al.* have given a $\frac{4}{3}$-competitive randomized knowledge state algorithm for this problem for cache size of 2 and have also proved a matching lower bound; thus this online algorithm is best possible. In addition, a lower bound of $\frac{3}{2}$ on the competitiveness for larger cache sizes was shown.

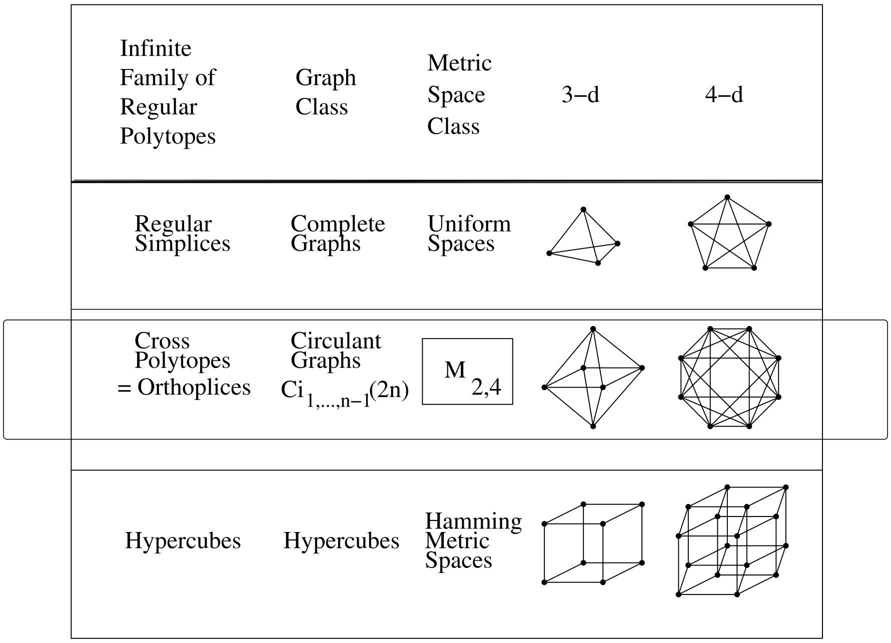

Fig. 4 The class $\mathcal{M}_{2,4}$ and its relation to uniform and hamming metric spaces

In this paper we have given an H_k-competitive randomized online algorithm for the k-paging problem which keeps track of only $3k$ pages. For large k, Lemma 15 can be improved to αk where $\alpha \approx 2.2572$ satisfies $\alpha^2 - \alpha - \alpha \ln(\alpha) = 1$.

For $k = 2$, we have shown that keeping track of three pages (*i.e.*, using one bookmark) suffices to obtain an optimally competitive randomized algorithm, and for $k = 3$, we have shown that keeping track of five pages (*i.e.*, using two bookmarks) suffices. We emphasize that we have not proven, nor do we believe, $3k$ to be the minimum number of pages needed for any particular k. In fact, we conjecture that a stronger result holds than the upper bound from this paper:

Conjecture 1 *There exists a randomized online algorithm for paging which is H_k-competitive and uses $o(k)$ bookmarks.*

Acknowledgement We thank the two referees for many helpful comments, which have improved the paper.

References

1. Achlioptas, D., Chrobak, M., Noga, J.: Competitive analysis of randomized paging algorithms. Theor. Comput. Sci. **234**, 203–218 (2000)
2. Bartal, Y., Chrobak, M., Larmore, L.L.: A randomized algorithm for two servers on the line. In: Proceedings 6th European Symposium on Algorithms (ESA). Lecture Notes in Computer Science, pp. 247–258. Springer, Berlin (1998)

3. Bartal, Y., Chrobak, M., Larmore, L.L.: A randomized algorithm for two servers on the line. Inf. Comput. **158**, 53–69 (2000)
4. Bein, W., Fleischer, R., Larmore, L.L.: Limited bookmark randomized online algorithms for the paging problem. Inf. Process. Lett. **76**, 155–162 (2000)
5. Bein, W., Larmore, L.L.: Trackless online algorithms for the server problem. Inf. Process. Lett. **74**, 73–79 (2000)
6. Bein, W., Larmore, L.L.: Trackless and limited bookmark algorithms for paging. SIGACT News **35**, 40–49 (2004)
7. Bein, W., Larmore, L.L., Reischuk, R.: Knowledge states for the caching problem in shared memory multiprocessor systems. Int. J. Found. Comput. Sci. **40**(1), 167–183 (2009)
8. Bein, W.W., Iwama, K., Kawahara, J., Larmore, L.L., Oravec, J.A.: A randomized algorithm for two servers in cross polytope spaces. In: Proceedings of 5th Workshop on Approximation and Online Algorithms (WAOA), Eilat, Israel, October 11–12, 2007. Lecture Notes in Computer Science, vol. 4927, pp. 246–259. Springer, Berlin (2008)
9. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
10. Chrobak, M., Koutsoupias, E., Noga, J.: More on randomized on-line algorithms for caching. Theor. Comput. Sci. **290**, 1997–2008 (2003)
11. Chrobak, M., Larmore, L.L.: The server problem and on-line games. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 7, pp. 11–64 (1992)
12. Coppersmith, D., Doyle, P.G., Raghavan, P., Snir, M.: Random walks on weighted graphs and applications to online algorithms. In: Proceedings 22nd Symposium on Theory of Computing (STOC), pp. 369–378. ACM, New York (1990)
13. Fiat, A., Karp, R., Luby, M., McGeoch, L.A., Sleator, D., Young, N.E.: Competitive paging algorithms. J. Algorithms **12**, 685–699 (1991)
14. Koutsoupias, E., Papadimitriou, C.: Beyond competitive analysis. In: Proceedings 35th Symposium on Foundations of Computer Science (FOCS), pp. 394–400. IEEE, New York (1994)
15. Koutsoupias, E., Papadimitriou, C.: Beyond competitive analysis. SIAM J. Comput. **30**, 300–317 (2000)
16. Schläfli, L.: Theorie der vielfachen Kontinuität. Birkhäuser, Basel (1857)