Knowledge States: A Tool in Randomized Online Algorithms

Wolfgang Bein

Center for the Advanced Study of Algorithms
School of Computer Science
University of Nevada, Las Vegas

ADS 2007

coauthors: Lawrence L. Larmore, John Noga, Rüdiger Reischuk

supported by NSF grant CCR-0312093
**offline**: all input data is completely available before the algorithm starts.
**Online Problems**

**offline**: all input data is completely available before the algorithm starts.

**online**: input data arrives a piece at a time the algorithm must make a decision without knowledge of the entire input.
**Online Problems**

*offline*: all input data is completely available before the algorithm starts.

*online*: input data arrives a piece at a time; the algorithm must make a decision without knowledge of the entire input.

*online problems*: resource allocation in operating systems, network routing, robotics, data-structuring, distributed computing, scheduling...
It is difficult to construct good randomized online algorithms

Use of work functions in the context of randomized online algorithms
Theme of the Talk

It is difficult to construct good randomized online algorithms

Use of work functions in the context of randomized online algorithms

Give distributional descriptions of algorithms
Theme of the Talk

It is difficult to construct good randomized online algorithms

Use of work functions in the context of randomized online algorithms

Give distributional descriptions of algorithms

Use the concept of forgiveness
It is difficult to construct good randomized online algorithms

Use of work functions in the context of randomized online algorithms

Give distributional descriptions of algorithms

Use the concept of forgiveness

Give two new results:

- better than 2-competitive 2-server algorithm in cross polytope spaces
- optimally competitive k-paging with only $O(k)$ memory
Tutorial at www.cs.unlv.edu/~bein/tutorial
Paging

- Universe of pages in “slow” memory
- Fast memory can hold $k$ pages
- Requests $\rho = r_1 r_2 \ldots r_n$ for pages have to be served
- Hit (no cost) or Miss (cost 1)

Fast Memory

```
| a | b | f | h | j | k |
```

Fast Memory

```
| x | a | b | f | j | k |
```

Eject h

Request $x$

Solution can be described as a sequence of configurations
The CNN Problem

- News crew in Manhattan (streets and avenues)
- “event” sequence $\varrho = r^1 r^2, \ldots, r^n$
- Event can be “seen” either horizontally or vertically
- Solution can be described as a sequence configurations
- **Goal:** minimize total movement cost
The CNN Problem

- News crew in Manhattan (streets and avenues)
- “event” sequence
  \[ \varrho = r_1 r_2, \ldots, r_n \]
- Event can be “seen” either horizontally or vertically
- Solution can be described as a sequence configurations
- **Goal:** minimize total movement cost
The CNN Problem

- News crew in Manhattan (streets and avenues)
- “event” sequence \( \psi = r_1^1 r_2^2, \ldots, r_n^n \)
- Event can be “seen” either horizontally or vertically
- Solution can be described as a sequence configurations
- **Goal**: minimize total movement cost
The CNN Problem

- News crew in Manhattan (streets and avenues)
- "event" sequence \( \varrho = r^1 r^2, \ldots, r^n \)
- Event can be "seen" either horizontally or vertically
- Solution can be described as a sequence configurations
- **Goal:** minimize total movement cost
The CNN Problem

- News crew in Manhattan (streets and avenues)
- “event” sequence \( \varrho = r^1 r^2, \ldots, r^n \)
- Event can be “seen” either horizontally or vertically
- Solution can be described as a sequence configurations
- **Goal**: minimize total movement cost
The CNN Problem

- News crew in Manhattan (streets and avenues)
- "event" sequence
  \[ \varrho = r_1, r_2, \ldots, r_n \]
- Event can be "seen" either horizontally or vertically
- Solution can be described as a sequence configurations
- **Goal:** minimize total movement cost
The CNN Problem

- News crew in Manhattan (streets and avenues)
- "event" sequence \( \varrho = r^1 r^2, \ldots, r^n \)
- Event can be "seen" either horizontally or vertically
- Solution can be described as a sequence configurations
- **Goal:** minimize total movement cost
The CNN Problem

- News crew in Manhattan (streets and avenues)
- “event” sequence $\varrho = r_1 r^2, \ldots, r^n$
- Event can be “seen” either horizontally or vertically
- Solution can be described as a sequence configurations
- **Goal:** minimize total movement cost
The 2-Server Problem

- 2 servers in a metric space $M$
- request sequence $\varrho = r^1 r^2, \ldots, r^n$
- online: decision must be made before $r^{i+1}$ is revealed
- **Goal**: minimize total movement cost
The 2-Server Problem

- 2 servers in a metric space $M$
- request sequence $\rho = r_1 r_2, \ldots, r_n$
- online: decision must be made before $r^{i+1}$ is revealed
- **Goal:** minimize total movement cost
The 2-Server Problem

- 2 servers in a metric space $M$
- request sequence $\rho = r^1 r^2, \ldots, r^n$
- online: decision must be made before $r^{i+1}$ is revealed
- **Goal:** minimize total movement cost
The 2-Server Problem

- 2 servers in a metric space $M$
- request sequence $\varrho = r^1 r^2, \ldots, r^n$
- online: decision must be made before $r^{i+1}$ is revealed
- **Goal:** minimize total movement cost
The 2-Server Problem

- 2 servers in a metric space $M$
- request sequence $\varrho = r^1r^2, \ldots, r^n$
- online: decision must be made before $r^{i+1}$ is revealed
- **Goal:** minimize total movement cost
The 2-Server Problem

- 2 servers in a metric space $M$
- request sequence $\mathcal{q} = r^1 r^2, \ldots, r^n$
- online: decision must be made before $r^{i+1}$ is revealed
- **Goal**: minimize total movement cost
The 2-Server Problem

- 2 servers in a metric space $M$
- request sequence $\varrho = r^1 r^2, \ldots, r^n$
- online: decision must be made before $r^{i+1}$ is revealed
- **Goal:** minimize total movement cost
The 2-Server Problem

- 2 servers in a metric space $M$
- request sequence $\rho = r^1 r^2, \ldots, r^n$
- online: decision must be made before $r^{i+1}$ is revealed
- **Goal:** minimize total movement cost
The 2-Server Problem

- 2 servers in a metric space $M$
- request sequence $\rho = r^1 r^2, \ldots, r^n$
- online: decision must be made before $r^{i+1}$ is revealed
- **Goal**: minimize total movement cost
the locations of the two servers is called a configuration.

A solution can be described as a sequence of configurations.

The movement cost is the transportation distance between configurations.
the locations of the two servers is called a configuration

solution can be described as a sequence of configurations

the movement cost is the transportation distance between configurations
the locations of the two servers is called a configuration

solution can be described as a sequence of configurations

the movement cost is the transportation distance between configurations
the locations of the two servers is called a configuration

solution can be described as a sequence of configurations

the movement cost is the transportation distance between configurations
the locations of the two servers is called a configuration

solution can be described as a sequence of configurations

the movement cost is the transportation distance between configurations
Algorithm $\mathcal{A}$ is at some initial configuration $a^0$
1. Algorithm $A$ is at some initial configuration $a^0$.

2. Requests: $\rho = r^1, \ldots, r^n$. 
Algorithm $A$ is at some initial configuration $a^0$.

Requests: $\rho = r^1, \ldots, r^n$.

At time $(t - 1)$, $A$ is at configuration $a^{t-1}$. 
Online Algorithms

1. Algorithm $\mathcal{A}$ is at some initial configuration $a^0$.
2. Requests: $\varrho = r^1, \ldots, r^n$.
3. At time $(t - 1)$, $\mathcal{A}$ is at configuration $a^{t-1}$.
4. $\mathcal{A}$ has to serve $r^t$ not knowing $r^{t+1}, \ldots$. 

Knowledge States: A Tool in Randomized Online Algorithms
Online Algorithms

1. Algorithm $\mathcal{A}$ is at some initial configuration $a^0$
2. Requests: $\varrho = r^1, \ldots, r^n$.
3. At time $(t - 1)$, $\mathcal{A}$ is at configuration $a^{t-1}$.
4. $\mathcal{A}$ has to serve $r^t$ not knowing $r^{t+1}, \ldots$
5. $\mathcal{A}$ chooses a configuration $a^t$. 
Algorithm $A$ is at some initial configuration $a^0$

2 Requests: $\rho = r^1, \ldots, r^n$.

3 At time $(t - 1)$, $A$ is at configuration $a^{t-1}$.

4 $A$ has to serve $r^t$ not knowing $r^{t+1}, \ldots$.

5 $A$ chooses a configuration $a^t$.

6 $A$ incurs $cost(a^{t-1}, r^t, a^t)$. 
Algorithm $\mathcal{A}$ is at some initial configuration $a^0$

2 Requests: $\rho = r^1, \ldots, r^n$.

3 At time $(t - 1)$, $\mathcal{A}$ is at configuration $a^{t-1}$.

4 $\mathcal{A}$ has to serve $r^t$ not knowing $r^{t+1}, \ldots$

5 $\mathcal{A}$ chooses a configuration $a^t$.

6 $\mathcal{A}$ incurs $cost(a^{t-1}, r^t, a^t)$.

If $\mathcal{A}$ uses randomization in bullet 5 then $\mathcal{A}$ is called a randomized online algorithm.
Example: $k = 2$ and $\rho = xyxyz$

```
Example 2
y 2
1
x 2
x

y
z

y
x
z
```

cost = 7

“Work Function Algorithm” (WFA)
Function on configurations: Dynamic programming

The optimal cost of being **there then**

\[ \text{optcost} = 4 = \min \text{last workfunction} \]

**Given request sequence** \( \rho \)

\[ \omega^\rho(a) = \min \text{cost of serving } \rho \text{ and ending in configuration } a \in X \]
Support of a Work Functions

\[
\begin{array}{|c|c|c|c|}
\hline
& \omega(\{y,z\}) & \omega(\{x,z\}) & \omega(\{x,y\}) \\
\hline
\text{initial} & 0 & 1 & 2 \\
\text{request } x & 2 & 1 & 2 \\
\text{request } y & 2 & 3 & 2 \\
\text{request } x & 4 & 3 & 2 \\
\text{request } y & 4 & 4 & 2 \\
\text{request } z & 4 & 4 & 6 \\
\hline
\end{array}
\]

\(S \subseteq \mathcal{X}\) supports \(\omega\) if for any \(b \in \mathcal{X}\) there exists some \(a \in S\) such that \(\omega(b) = \omega(a) + |a, b|\).
Support of a Work Functions

\[\omega(\{y, z\}) \quad \omega(\{x, z\}) \quad \omega(\{x, y\})\]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>request x</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>request y</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>request z</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

\[S \subseteq \mathcal{X} \text{ supports } \omega \text{ if for any } b \in \mathcal{X} \text{ there exists some } a \in S \text{ such that } \omega(b) = \omega(a) + |a, b|\.

A “reasonable algorithm” will move to configurations in the support.
Support of a Work Functions

<table>
<thead>
<tr>
<th></th>
<th>$\omega({y, z})$</th>
<th>$\omega({x, z})$</th>
<th>$\omega({x, y})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>request x</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>request y</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>request x</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>request y</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>request z</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

$S \subseteq \mathcal{X}$ supports $\omega$ if for any $b \in \mathcal{X}$ there exists some $a \in S$ such that $\omega(b) = \omega(a) + |a, b|$.

A “reasonable algorithm” will move to configurations in the support.

WFA moves for request $r$ from configuration $a$ to configuration $b$ such that $|a, b| + \omega(b)$ is minimized.
Competitiveness

For request sequence $\varrho = r^1, r^2, \ldots$ consider

\[ \text{cost}_A(\varrho): \text{the cost on } \varrho \text{ achieved by } A \]
\[ \text{cost}_{opt}(\varrho): \text{the cost on } \varrho \text{ achieved by } opt \]

We say that $A$ is $C$-competitive if for each sequence $\varrho$ we have

\[ E \text{cost}_A(\varrho) \leq C \cdot \text{cost}_{opt}(\varrho) + K \]

Example:

\[ \frac{\text{cost}_{WFA}(xyxyz)}{\text{cost}_{opt}(xyxyz)} = \frac{7}{4} \quad \text{WFA is 2-competitive} \]
The Distributional Model

A randomized algorithm can be viewed as a deterministic algorithm on distributions.

\[ \mathcal{X} = \text{all configurations} \]
\[ \pi \text{ is a distribution on } \mathcal{X}. \]

- Algorithm \( \mathcal{A} \) is at some initial configuration \( \mathbf{a}^0 \).
A randomized algorithm can be viewed as a deterministic algorithm on distributions.

\[ \mathcal{X} = \text{all configurations} \]

\[ \pi \text{ is a distribution on } \mathcal{X}. \]

- Algorithm \( \mathcal{A} \) is at some initial configuration \( a^0 \).
- Requests: \( \varrho = r^1, \ldots, r^n \).
A randomized algorithm can be viewed as a deterministic algorithm on distributions.

\[ \mathcal{X} = \text{all configurations} \]
\[ \pi \text{ is a distribution on } \mathcal{X}. \]

- Algorithm $\mathcal{A}$ is at some initial configuration $a^0$.
- Requests: $\varrho = r^1, \ldots, r^n$.
- At time $(t - 1)$, $\mathcal{A}$ is at distribution $\pi^{t-1}$. 
A randomized algorithm can be viewed as a deterministic algorithm on distributions.

\[ \mathcal{X} = \text{all configurations} \]
\[ \pi \text{ is a distribution on } \mathcal{X}. \]

- Algorithm \( \mathcal{A} \) is at some initial configuration \( a^0 \).
- Requests: \( Q = r^1, \ldots, r^n \).
- At time \( (t - 1) \), \( \mathcal{A} \) is at distribution \( \pi^{t-1} \).
- \( \mathcal{A} \) has to serve \( r^t \) not knowing \( r^{t+1}, \ldots \).
A randomized algorithm can be viewed as a deterministic algorithm on distributions.

$$X = \text{all configurations}$$

$$\pi$$ is a distribution on $$X$$.

- Algorithm $$A$$ is at some initial configuration $$a^0$$.
- Requests: $$\rho = r^1, \ldots, r^n$$.
- At time $$(t - 1)$$, $$A$$ is at distribution $$\pi^{t-1}$$.
- $$A$$ has to serve $$r^t$$ not knowing $$r^{t+1}, \ldots$$
- $$A$$ chooses deterministically a distribution $$\pi^t$$. 
The cost incurred by moving from one distribution to the next is calculated by moving mass along a transportation problem.

The transportation problem has the Monge property.
Problem: The “support” grows without bound.
Forgiveness

Lower the work function on selective configurations
Forgiveness

Lower the work function on selective configurations
Forgiveness

Lower the work function on selective configurations

Work functions are now estimators
Algorithm is constructed using the “mixed model” of online computation
The Randomized 2-Server Problem

Best: RANDOM SLACK 2-competitive
[Coppersmith, Doyle, Raghavan, Snir, 90]
The Randomized 2-Server Problem

- Best: RANDOM SLACK 2-competitive
  [Coppersmith, Doyle, Raghavan, Snir, 90]
- Not known to be best possible.
The Randomized 2-Server Problem

- Best: RANDOM SLACK 2-competitive
  [Coppersmith, Doyle, Raghavan, Snir, 90]
- Not known to be best possible.
- Lower Bound: $1 + e^{-\frac{1}{2}} \approx 1.6065$
  [Chrobak, Larmore, Lund, Reingold, 97]
The Randomized 2-Server Problem

- Best: RANDOM SLACK 2-competitive  
  [Coppersmith, Doyle, Raghavan, Snir, 90]
- Not known to be best possible.
- Lower Bound: $1 + e^{-\frac{1}{2}} \approx 1.6065$  
  [Chrobak, Larmore, Lund, Reingold, 97]
- Line: $\frac{155}{78} \approx 1.987$  
  [Bartal, Chrobak, Larmore, 98]
2-Server Problem: $\mathcal{M}_{24}$

$\mathcal{M}_{24}$ consists of all metric spaces such that
- All distances are 1 or 2.
- $d(x, y) + d(x, z) + d(y, z) \leq 4$
Why $M_{24}$?

- A step in the direction of the goal (better than 2-competitive randomized algorithm for the 2-server problem).
Why $M_{24}$?

- A step in the direction of the goal (better than 2-competitive randomized algorithm for the 2-server problem).
- Allows a simple example of the knowledge state method.
Why $M_{24}$?

- A step in the direction of the goal (better than 2-competitive randomized algorithm for the 2-server problem).
- Allows a simple example of the knowledge state method.
- An interesting class in its own right, generalizing the octahedron.
Exactly Three Infinite Families of Convex Regular Polytopes

(Ludwig Schlafli, 1852)

<table>
<thead>
<tr>
<th>Infinite Family of Regular Polytopes</th>
<th>Graph Class</th>
<th>Metric Space Class</th>
<th>3-d</th>
<th>4-d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Simplices</td>
<td>Complete Graphs</td>
<td>Uniform Spaces</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross Polytopes</td>
<td>Circulant Graphs</td>
<td>M_{24}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>= Orthoplices</td>
<td>Ci_{1,...,n-1}(2n)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hypercubes</td>
<td>Hypercubes</td>
<td>Hamming Metric Spaces</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is a Knowledge State?

Knowledge state \( k = (\omega, \pi) \):

- \( \omega : \mathcal{X} \to \mathbb{R} \) is the estimator.
- \( \pi \) is a distribution on \( \mathcal{X} \).

\[ \pi(x, y) \] is the probability we are at \( \{x, y\} \).

\[ \omega(x, y) \] is the estimated unpaid cost of the adversary if it is at \( \{x, y\} \).
A Closer Look

The estimator and distribution are defined for all configurations but characterized by their values only on the support

If \( a \in \mathcal{X} - S \), then

- \( \pi(a) = 0 \).
- \( \omega(a) = \min_{b \in S} \{ \omega(b) + \|a, b\| \} \)
Up to symmetry, there are 8 knowledge states of a $\frac{19}{12}$-competitive algorithm for $\mathcal{M}_{2,4}$.
There are Numerous Moves. Here is One.
One Move of the Algorithm

- Start at a standard knowledge state over \((x, y, z)\).
One Move of the Algorithm

- Start at a **standard** knowledge state over \((x, y, z)\).
- Read a request \(r\).
One Move of the Algorithm

- Start at a **standard** knowledge state over \((x, y, z)\).
- Read a request \(r\).
- Update the estimator.
One Move of the Algorithm

- Start at a **standard** knowledge state over \((x, y, z)\).
- Read a request \(r\).
- Update the estimator.
- Move the distribution.
One Move of the Algorithm

- Start at a **standard** knowledge state over \((x, y, z)\).
- Read a request \(r\).
- Update the estimator.
- Move the distribution.
- **Las Vegas.** Randomly pick a subsequent.
One Move of the Algorithm

- Start at a **standard** knowledge state over \((x, y, z)\).
- Read a request \(r\).
- Update the estimator.
- Move the distribution.
- **Las Vegas.** Randomly pick a **subsequent**.
- We are at a **standard** knowledge state over \((x, y, r)\), \((y, x, r)\), \((x, z, r)\), \((z, x, r)\), \((y, z, r)\), or \((z, y, r)\).
Behavioral Version: The Wireframe Algorithm
Results for the 2-Server Problem in $\mathcal{M}_{2,4}$

- 2-Server Problem on $\mathcal{M}_{2,4}$, $C = \frac{7}{4}$
Results for the 2-Server Problem in $\mathcal{M}_{2,4}$

- 2-Server Problem on $\mathcal{M}_{2,4}$, $C = \frac{7}{4}$
- 2-Server Problem on $\mathcal{M}_{2,4}$, $C = \frac{19}{12} \approx 1.583$
Results for the 2-Server Problem in $\mathcal{M}_{2,4}$

- 2-Server Problem on $\mathcal{M}_{2,4}$, $C = \frac{7}{4}$
- 2-Server Problem on $\mathcal{M}_{2,4}$, $C = \frac{19}{12} \approx 1.583$
- This is optimal for $\mathcal{M}_{2,4}$. 
Results for the 2-Server Problem in $\mathcal{M}_{2,4}$

- 2-Server Problem on $\mathcal{M}_{2,4}$, $C = \frac{7}{4}$
- 2-Server Problem on $\mathcal{M}_{2,4}$, $C = \frac{19}{12} \approx 1.583$
- This is optimal for $\mathcal{M}_{2,4}$.
- Uniform spaces i.e. paging, the optimal competitiveness is $C = 1.5$. 
Results for the 2-Server Problem in $\mathcal{M}_{2,4}$

- 2-Server Problem on $\mathcal{M}_{2,4}$, $C = \frac{7}{4}$
- 2-Server Problem on $\mathcal{M}_{2,4}$, $C = \frac{19}{12} \approx 1.583$
- This is optimal for $\mathcal{M}_{2,4}$.
- Uniform spaces i.e. paging, the optimal competitiveness is $C = 1.5$.
- Open: a better than 2-competitive randomized algorithm for 2 servers in general spaces.
Paging

Paging: k-server problem in uniform spaces

- CNN
- Amazon
- Yahoo
- UNLV
- Sandia

..requesting NYTimes
History of k-paging

- [Fiat, Karp, Luby, McGeoch, Sleator, Young, 1991]
  - Lower bound, $H_k = \sum_{i=1}^{k} 1/i$
  - $(2H_k - 1)$-competitive algorithm “RMARK”
  - RMARK uses $O(n)$ memory
History of k-paging

- [Fiat, Karp, Luby, McGeoch, Sleator, Young, 1991]
  - Lower bound, $H_k = \sum_{i=1}^{k} 1/i$
  - $(2H_k - 1)$-competitive algorithm “RMARK”
  - RMARK uses $O(n)$ memory

- [McGeoch, Sleator, 1991]
  - $H_k$-competitive algorithm PARTITION
  - unbounded memory
History of k-paging

- [Fiat, Karp, Luby, McGeoch, Sleator, Young, 1991]
  - Lower bound, $H_k = \sum_{i=1}^{k} 1/i$
  - $(2H_k - 1)$-competitive algorithm “RMARK”
  - RMARK uses $O(n)$ memory

- [McGeoch, Sleator, 1991]
  - $H_k$-competitive algorithm PARTITION
  - unbounded memory

- [Achlioptas, Chrobak, Noga, 2000]
  - $H_k$-competitive algorithm EQUITABLE
  - $O(k^2 \log k)$ memory
History of k-paging

- [Fiat, Karp, Luby, McGeoch, Sleator, Young, 1991]
  - Lower bound, $H_k = \sum_{i=1}^{k} 1/i$
  - $(2H_k - 1)$-competitive algorithm “RMARK”
  - RMARK uses $O(n)$ memory

- [McGeoch, Sleator, 1991]
  - $H_k$-competitive algorithm PARTITION
  - unbounded memory

- [Achlioptas, Chrobak, Noga, 2000]
  - $H_k$-competitive algorithm EQUITABLE
  - $O(k^2 \log k)$ memory

- [Bein, Larmore, Noga, 2007]
  - $H_k$-competitive algorithm EQUITABLE2
  - $O(k)$ memory
A trackless algorithm (i.e. an algorithm that does not use any bookmarks) cannot have optimal competitiveness. [Bein, Fleischer, Larmore, 00]
Algorithm $K_2$:

Competitiveness: $C_{K_2} = \frac{3}{2}$
Behavioral

Distributional

Knowledge States: A Tool in Randomized Online Algorithms
Work Functions and Offset Functions

Knowledge States: A Tool in Randomized Online Algorithms
EQUITABLE [Achlioptas, Chrobak, Noga, 2000]

- The algorithm is described using the distributional model
- The algorithm’s distribution mass is only on the support of the work function

```
1 1/2 1/2
1/3 1/3
1/4 1/4 1/4
```
Algorithm is constructed using the “mixed model” of online computation
Knowledge States: A Tool in Randomized Online Algorithms
Work functions are denoted using the “bar notation”.

A tuple $T$ is in the support: at least $i$ members of $T$ are to the left of the $i^{th}$ bar.
The Case $k = 3$

Knowledge States: A Tool in Randomized Online Algorithms
For general $k$:

- **EQUITABLE** forgives to a “cone” after $O(k^2 \log k)$ bookmarks
For general $k$:

- EQUITABLE forgives to a “cone” after $O(k^2 \log k)$ bookmarks
- EQUITABLE forgives to a work function with small support (but not a cone) after $O(k)$ bookmarks
Further Research

- The 2-server problem for general spaces
The 2-server problem for general spaces

CNN:

Deterministic: lower bound: $6 + \sqrt{17}$
[Koutsoupias, Taylor, 2005]
Deterministic: upper bound: $10^5, 879$
[Sitters Stougie 2005]
Randomized: Open