

A Randomized Algorithm for Two Servers in Cross Polytope Spaces

Wolfgang Bein¹, Kazuo Iwama², Jun Kawahara², Lawrence L. Larmore¹, and James A. Oravec¹

¹ Center for the Advanced Study of Algorithms, School of Computer Science, University of Nevada Las Vegas, Nevada 89154, USA. **

bein@cs.unlv.edu larmore@cs.unlv.edu oravec@cs.unlv.edu

² School of Informatics, Kyoto University, Kyoto 606-8501, Japan.

iwama@kuis.kyoto-u.ac.jp jkawahara@kuis.kyoto-u.ac.jp

Abstract. It has been a long-standing open problem to determine the exact randomized competitiveness of the 2-server problem, that is, the minimum competitiveness of any randomized online algorithm for the 2-server problem. For deterministic algorithms the best competitive ratio that can be obtained is 2 and no randomized algorithm is known that improves this ratio for general spaces. For the line, Bartal *et al.* [2] give a $\frac{155}{78}$ competitive algorithm, but their algorithm is specific to the geometry of the line.

We consider here the 2-server problem over Cross Polytope Spaces $M_{2,4}$. We obtain an algorithm with competitive ratio of $\frac{19}{12}$, and show that this ratio is best possible. This algorithm gives the second non-trivial example of metric spaces with better than 2 competitive ratio.

The algorithm uses a design technique called the knowledge state technique – a method not specific to $M_{2,4}$.

1 Background

In the k -server problem, there are k mobile identical servers in a metric space \mathcal{M} . At any time, a point $r \in M$ can be “requested,” and must be “served” by moving one of the k servers to the point r . The cost of that service is defined to be the distance the server is moved; for a sequence of requests the goal is to serve the requests at small cost. An *online algorithm* for the server problem decides, at each request, which server to move, but does not know the sequence of future requests. We analyze an online algorithm for the server problem in terms of its *competitive ratio*, which essentially gives the ratio of its cost over the cost of an optimal (offline) algorithm which has knowledge of the entire request sequence before making any decisions. More precisely, we say that an online algorithm \mathcal{A} for the server problem is C -competitive, if there is a constant K , such that, given

** Research of the first author (Bein) done while visiting Kyoto University as Kyoto University Visiting Professor. Research of the first author (Bein) and the fourth author (Larmore) supported by NSF grant CCR-0312093.



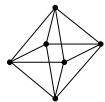
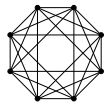
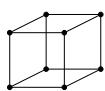
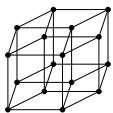
Infinite Family of Regular Polytopes	Graph Class	Metric Space Class	3-d	4-d
Regular Simplices	Complete Graphs	Uniform Spaces		
Cross Polytopes = Orthoplices	Circulant Graphs $Ci_{1, \dots, n-1}(2n)$	M_{24}		
Hypercubes	Hypercubes	Hamming Metric Spaces		

Fig. 1: The Class $M_{2,4}$

any request sequence ϱ , $cost_{\mathcal{A}}(\varrho) \leq C \cdot cost_{OPT}(\varrho) + K$. For a randomized online algorithm, we state competitiveness in terms of expected cost. The competitive ratio of \mathcal{A} is the smallest C for which \mathcal{A} is C -competitive.

The server problem was first proposed by Manasse, McGeoch and Sleator [14] and the problem has been studied widely since then. They also introduced the now well-known *k-server conjecture*, which states that, for each k , there exists an online algorithm for k servers which is k -competitive in any metric space. The conjecture was immediately proved true for $k = 2$, but for larger k remains open except in special cases, including lines [8], trees [9], and spaces with at most $k + 2$ points [11]. Even some simple-looking special cases have not been settled, for example the 3-server problem in the circle and in the Euclidean plane [8, 9, 12]. In general, the best currently known upper bound is $2k - 1$, given by Koutsoupias and Papadimitriou [12]. Thus there is a rich literature for *deterministic* online algorithms for this problem.

Randomization is a powerful for many online problems [7]. Yet, very little is known for randomized algorithms for the k -server problem. It seems to be quite hard 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 and it is quite intuitive that a “better than 2-competitive” algorithm should exist. In fact, $1 + e^{-\frac{1}{2}} \approx 1.6065$, is the greatest lower bound with a published proof (see [10]) on the competitiveness of any randomized on-

line algorithm.³ There has been some progress for special cases. The randomized competitiveness is known to be $\frac{3}{2}$ for all uniform spaces and is also known for three-point spaces [13]. For the special case of the line, Bartal *et al.* [2] have given a randomized algorithm with a competitive ratio of $\frac{155}{78} \approx 1.987$.

Our Contribution. In this paper we give a randomized online algorithm for the 2-server problem in Cross Polytope Spaces with optimal competitive ratio of $\frac{19}{12}$. Cross Polytope Spaces, denoted by \mathcal{M}_{24} , have been studied extensively as early as the 19th century, see Schläfli [15], as well as Figure 1. They consist of all metric spaces such that

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

By an abuse of terminology we will sometimes simply say “the metric space $\mathcal{M}_{2,4}$ ” to denote this class of metric spaces.

In terms of the server problem, \mathcal{M}_{24} generalizes uniform spaces and thus paging. It is also useful to gain insight into the 2-server problem over more general spaces. Our technique can, in principle, be used to design algorithms for spaces $M_{\ell,k}$, $\ell \leq \frac{k}{2}$, where distances are $1, \dots, \ell$ and the perimeter of every triangle is at most k .

Our algorithm is not derived in an ad hoc way, instead it is constructed by using a design technique called the knowledge state technique. It is worth mentioning that it would be hard to come up with the actual behavioral algorithm, which we call the “wireframe algorithm”, if one were not to use this technique. Yet the algorithm can be easily implemented and uses little memory, though the derivation and the proof of competitiveness is only via the technique.

In the next section we briefly describe the knowledge state technique, and then give a knowledge state description of the algorithm together with a proof of competitiveness. This description is in what is called the mixed model of computation – a generalization of a distributional description of a randomized online algorithm. As mentioned, the technique makes it easier to contrive the $\frac{19}{12}$ -competitive algorithm. In this form however, the algorithm would be hard to implement as it is not described in the usual behavioral way. Thus in Section 3 we translate this description into the behavioral (and easily implementable) wireframe algorithm. We are also able to show that our algorithm has a competitive ratio, which is best possible; we show the lower bound in Section 4.

2 Knowledge States

We remind the reader that many randomized algorithms are given in distributional form, including a number of well known paging algorithms, *e.g.* [1, 3]. For the 2-server problem, such an algorithm is essentially a state transition diagram, where each state is a probabilistic distribution of configurations (each configuration is a set of two points in the space – the locations of the servers); a

³ A lower bound very slightly larger than $1 + e^{-\frac{1}{2}}$ is given in [10], but without proof.

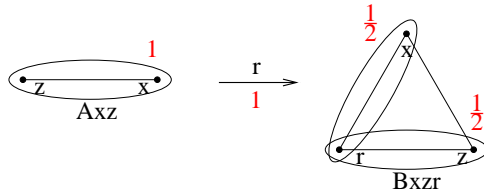


Fig. 2: A Step of a Distributional Algorithm

transition from one state to the next state is a deterministic transition to a new distribution. Figure 2 illustrates such a step. Here the algorithm has both servers initially at configuration (x, z) . Serving request r the algorithm transitions to a distribution with mass $\frac{1}{2}$ at (r, x) and mass $\frac{1}{2}$ at (r, z) . Unfortunately the number of configurations in each state (and hence the number of states) can increase arbitrarily. One way to help avoid this is to allow non-deterministic transitions. We note that we have a great degree of freedom in designing our state transition diagram. As it turns out, our algorithm needs only eight states to achieve the optimal competitive ratio.

We describe our algorithm and the lower bound result in terms of knowledge states. We describe knowledge states briefly in this section and refer the reader to [5, 6] for a more detailed description of this concept. It incorporates non-deterministic transitions as well as estimates on the offline cost.

As mentioned above, we use a variation of the *distribution model* to describe our randomized algorithm. That is, at each step the state of the algorithm will be described by a probability distribution on the set of all possible configurations at that step. The distribution model is equivalent to the behavioral model for randomized online algorithms against an oblivious adversary; see, for example, [7]. In the standard distribution model, the algorithm deterministically chooses a distribution at each step, but in this paper we allow the algorithm to use randomization to choose the distribution. We call such a step a *Las Vegas Step*; the reader might preview Figure 4. This variation, called the *mixed model* of randomized algorithms, is a generalization of both the behavioral model and the distributional model.

Let \mathcal{X} denote the set of all configurations. (Naturally, for the 2-server problem, a configuration is simply a 2-tuple (a, b) of points in the metric space, which describes the location of two servers.) We say that a function $\omega : \mathcal{X} \rightarrow \mathbf{R}$ is *Lipschitz* if $\omega(w) \leq \omega(u) + d(u, w)$ for all $u, w \in \mathcal{X}$. An *estimator* is a non-negative Lipschitz function $\mathcal{X} \rightarrow \mathbf{R}$. If $S \subseteq \mathcal{X}$, we say that S *supports an estimator* ω if, for any $w \in \mathcal{X}$ there exists some $u \in S$ such that $\omega(w) = \omega(u) + d(u, w)$. If ω is supported by a finite set, then there is a unique minimal set S which supports ω , which we call the *estimator support* of ω . We call the cardinality of the support

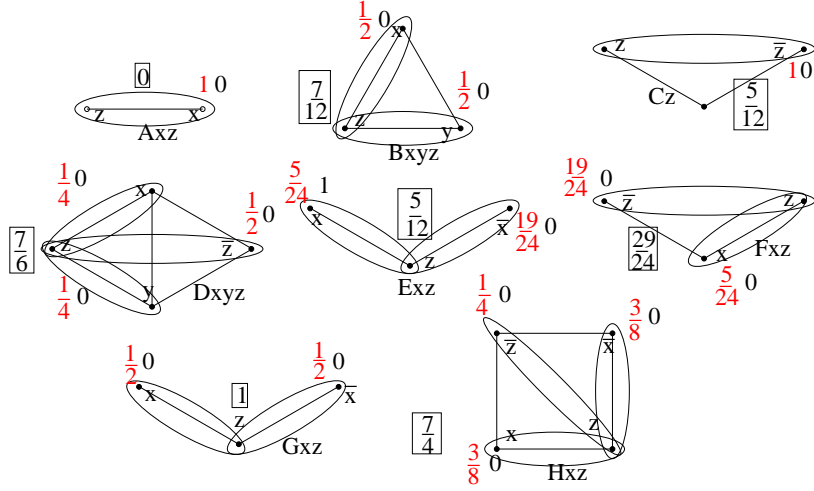


Fig. 3: The Knowledge States

the *order* of the estimator. We say that an estimator ω has *zero minimum* if $\min_{u \in \mathcal{X}} \omega(u) = 0$.⁴

A *knowledge state algorithm* [4–6] is a mixed online algorithm that computes an *estimator* at each step. The estimators used throughout this paper will have very low order, i.e. the estimator can be described by giving values on very few configurations. Furthermore, distributions of a knowledge state algorithm are only concentrated on the estimator support, i.e. they are zero on all configurations other than the configurations in the estimator support.

More formally, if \mathcal{A} is a knowledge-state algorithm, then:

1. At any given step, \mathcal{A} keeps track of a pair (ω, π) , where π is a finite distribution on \mathcal{X} , and $\omega : \mathcal{X} \rightarrow \mathbb{R}$ is the current estimator. The distribution is positive only on configurations which are in the support of the estimator ω . We call that pair the *current knowledge state*.
2. If $S = (\omega, \pi)$ is the knowledge state and the next request is r , then \mathcal{A} uses randomization to pick a new knowledge state $S' = (\omega', \pi')$.

We now describe Item 1 for our specific situation. Thus, let M be a metric space in the class \mathcal{M}_{24} . We will call a finite set of points $S \subset M$ a *constellation*. To define the knowledge states for our algorithm, we only need a total of eight constellations, where each constellation has no more than four points.

⁴ We remind the reader of the concept of a *work function*. Work functions are estimator functions. For example, work functions were used by Lund and Reingold [13] to describe an “opt-graph”, which describes all possible moves of an optimal adversary. In short, work functions provide information about the optimal cost of serving the past request sequence. For a request sequence ϱ , by $\omega^\varrho(u)$, we denote the minimum cost of serving ϱ and ending in configuration u .

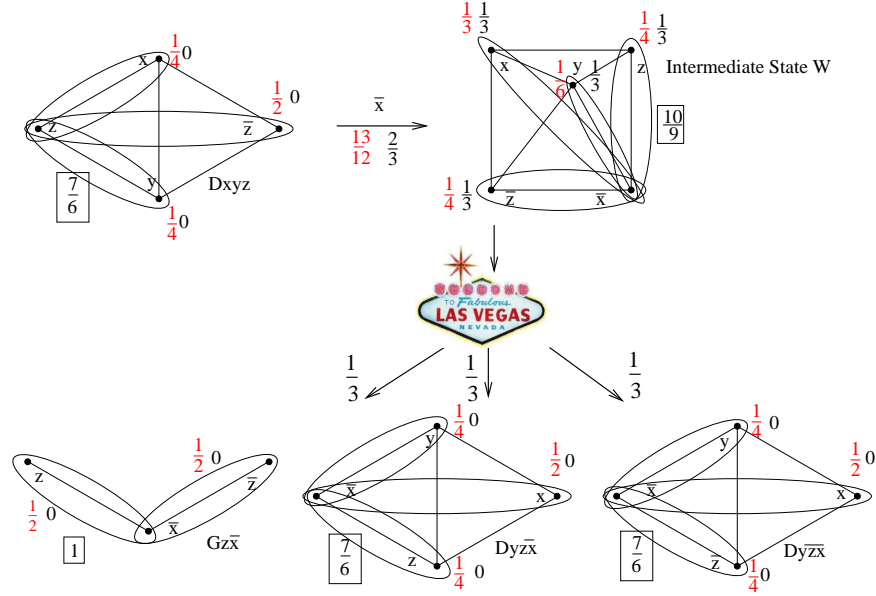


Fig. 4: One Move of the Knowledge State Algorithm

Each constellation is used to define a knowledge state of order no more than 3. In fact, for \mathcal{M}_{24} it will suffice to consider a very small and simple class of knowledge states: these knowledge states are shown in Figure 3. In Figure 3, a line between two points indicates a distance of 1 between the two points, and the absence of a line means that the points are 2 apart. Note also that for any point $x \in M$, we denote an antipodal point (*i.e.* a point a distance of 2 away) by \bar{x} . The ovals encircling two points are the support configurations of the estimators and distributions; the red numbers (the numbers to the left in the pairs of numbers) give the values of the distribution and the black numbers (the numbers to the right in the pairs of numbers) give the value on the support of the estimator. We refer to these knowledge states as $A_{xz}, B_{xyz}, C_z, D_{xyz}, E_{xz}, F_{xz}, G_{xz}$ and H_{xz} . When we only refer to the configurations we will use the same notation except we will use lower case letters; thus the constellations are referred to as $a_{xz}, b_{xyz}, c_z, d_{xyz}, e_{xz}, f_{xz}, g_{xz}$ and h_{xz} . We finally note that the numbers in the boxes denote a potential, which is used later.

We now turn to Item 2 and describe how, using randomization, a new knowledge state is chosen. Given $S = (\omega, \pi)$ and r there are subsequent knowledge states $S_i = (\omega_i, \pi_i)$ and subsequent nonactive weights λ_i for $i = 1, \dots, m$, $\sum_{i=1}^m \lambda_i = 1$. Then for each i , \mathcal{A} chooses S' to be S_i with probability λ_i . Again, for $\mathcal{M}_{2,4}$, Figure 3 shows all eight possible subsequents.

We will now discuss how we can see if a knowledge state algorithm is competitive. Given the subsequents, a real number $adjust_{\mathcal{A}}(S, r)$ is computed such that $(\omega \wedge r)(u) \geq adjust_{\mathcal{A}}(S, r) + \sum_{i=1}^m \lambda_i \omega_i(u)$ for each $x \in \mathcal{X}$, where we define func-

tion $\omega \wedge r$ as $(\omega \wedge r)(w) = \min \{\omega(u) + d(u, w) \mid u \ni r\}$. We will use a standard potential argument to prove competitiveness, and thus we will need to associate a potential Φ with each knowledge state. We now define the *update condition* for a given step. To this end, fix competitive ratio $C > 1$. Let S be the current knowledge state, let $\{S_i\}$ be the subsequents for the current step, and λ_i be the probability that S_i will be chosen in this current step. Let $cost_{\mathcal{A}}$ to be the expected cost of the algorithm \mathcal{A} . Then the update condition is that

$$\Phi(S) \geq cost_{\mathcal{A}} - C \cdot adjust + \sum_i \lambda_i \Phi(S_i). \quad (1)$$

We will make use of the following lemma from [5, 6]:

Lemma 1. *If the update condition holds at every step of a knowledge state algorithm then the algorithm is C -competitive.*

KS State	Request	Resulting KS	Φ_0	Φ_1	offset	$cost_{\mathcal{A}}$	slack
Axz	\bar{x}	Cx	0	$\frac{5}{12}$	1	1	$\frac{1}{6}$
Axz	r	$Bx zr$	0	$\frac{7}{12}$	1	1	0
$Bxyz$	x	Axz	$\frac{7}{12}$	0	0	$\frac{1}{2}$	$\frac{1}{18}$
$Bxyz$	\bar{z}	$Bxy\bar{z}$	$\frac{7}{12}$	$\frac{7}{12}$	1	1	$\frac{7}{12}$
$Bxyz$	\bar{x}	$Dyz\bar{x}$	$\frac{7}{12}$	$\frac{7}{6}$	1	1	0
$Bxyz$	r	$\frac{1}{3}Bxyr + \frac{1}{3}Bx zr + \frac{1}{3}By zr$	$\frac{7}{12}$	$\frac{7}{12}$	$\frac{2}{3}$	1	$\frac{1}{18}$
Cz	r	Gzr	$\frac{5}{12}$	1	1	1	0
$Dxyz$	x	$E\bar{z}x$	$\frac{7}{6}$	$\frac{5}{12}$	0	$\frac{3}{4}$	0
$Dxyz$	\bar{z}	Cz	$\frac{7}{6}$	$\frac{5}{12}$	0	$\frac{1}{2}$	$\frac{1}{4}$
$Dxyz$	\bar{x}	$\frac{1}{3}Dyz\bar{x} + \frac{1}{3}Dy\bar{z}\bar{x} + \frac{1}{3}Gz\bar{x}$	$\frac{7}{6}$	$\frac{10}{9}$	$\frac{2}{3}$	$\frac{13}{12}$	$\frac{1}{36}$
$Dxyz$	r	$\frac{1}{2}Bx zr + \frac{1}{2}By\bar{z}r$	$\frac{7}{6}$	$\frac{7}{12}$	$\frac{1}{2}$	1	$\frac{3}{8}$
Exz	x	Fzx	$\frac{5}{12}$	$\frac{29}{24}$	1	$\frac{19}{24}$	0
Exz	\bar{x}	$A\bar{x}z$	$\frac{5}{12}$	0	0	$\frac{5}{12}$	0
Exz	\bar{z}	$A\bar{x}z$	$\frac{5}{12}$	0	0	$\frac{5}{12}$	0
Exz	r	$B\bar{x}zr$	$\frac{5}{12}$	$\frac{7}{12}$	1	1	$\frac{5}{12}$
Fxz	x	Ezx	$\frac{29}{24}$	$\frac{5}{12}$	0	$\frac{19}{24}$	0
Fxz	\bar{z}	Cz	$\frac{29}{24}$	$\frac{5}{12}$	0	$\frac{5}{24}$	$\frac{7}{12}$
Fxz	\bar{x}	$Hz\bar{x}$	$\frac{29}{24}$	$\frac{7}{4}$	1	$\frac{25}{24}$	0
Fxz	r	$\frac{1}{2}Axr + \frac{1}{2}Gzr$	$\frac{29}{24}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{31}{24}$	$\frac{17}{24}$
Gxz	x	Axz	1	0	0	1	0
Gxz	\bar{z}	Cz	1	$\frac{5}{12}$	1	1	$\frac{7}{6}$
Gxz	r	Axr	1	0	$\frac{3}{2}$	1	$\frac{13}{12}$
Hxz	r	$\frac{1}{2}Bx zr + \frac{1}{2}B\bar{x}\bar{z}r$	$\frac{7}{4}$	$\frac{7}{12}$	$\frac{1}{2}$	1	$\frac{23}{24}$
Hxz	x	$E\bar{z}x$	$\frac{7}{4}$	$\frac{5}{12}$	0	1	$\frac{1}{3}$
Hxz	\bar{z}	Cz	$\frac{7}{4}$	$\frac{5}{12}$	0	$\frac{3}{4}$	$\frac{7}{12}$

Table 1: The Knowledge State Algorithm for $M_{2,4}$

Figure 4 shows the step where the knowledge state S is $Dxyz$ and \bar{x} is requested. For S we have an estimator with support $\omega(\{z, x\}) = \omega(\{z, y\}) = \omega(\{z, \bar{z}\}) = 0$ and distribution $\frac{1}{4}$ on $\{z, x\}$, $\frac{1}{4}$ on $\{z, y\}$ and $\frac{1}{2}$ on $\{z, \bar{z}\}$. In this situation the knowledge state algorithm chooses knowledge states $G_{z\bar{x}}$, $D_{yz\bar{x}}$, and $D_{y\bar{z}\bar{x}}$ with equal probability of $\frac{1}{3}$. (See the single numbers on the edges under the Las Vegas sign in Figure 4.) Next, we will argue that the update condition, *i.e.* inequality 1, does indeed hold for this step. To argue this we first focus on the “intermediate state” W depicted to the right of $Dxyz$. First note by using elementary arithmetic that the weighted average of the three subsequent states $G_{z\bar{x}}$, $D_{yz\bar{x}}$, and $D_{y\bar{z}\bar{x}}$ gives exactly the intermediate state W , both its distribution as well as its estimator function.

Turning now to $\omega \wedge \bar{x}$ it is easily calculated that the resulting estimator support set consists of $\{\{\bar{x}, x\}, \{\bar{x}, \bar{z}\}, \{\bar{x}, z\}, \{\bar{x}, y\}\}$ with value 1 on all the elements in the support set. Note now that if $\omega \wedge \bar{x}$ is lowered by $\frac{2}{3}$, this function is equal to the estimator of W . In other words, if $adjust_{\mathcal{A}}(S, \bar{x}) = \frac{2}{3}$, then $(\omega \wedge \bar{x}) - adjust_{\mathcal{A}}(S, \bar{x})$ is the estimator of W . (The value $adjust_{\mathcal{A}}(S, \bar{x})$ appears as the second value under the arrow from D_{xyz} to W in Figure 4.)

We now analyze the cost of the algorithm for the step. It is the cost of the move from the distribution of $Dxyz$ to the distribution of W . We remind the reader that this can be done solving a transportation problem. An instance of the *transportation problem* is a weighted directed bipartite graph with distributions on both parts. More formally, an instance is an ordered quintuple $(A, B, cost, \alpha, \beta)$ where U and V are finite non-empty sets, α is a distribution on U , β is a distribution on V , and $cost$ is a real-valued function on $U \times V$. A *solution* to this instance is a distribution γ on $U \times V$ such that

1. $\gamma(\{u\} \times V) = \alpha(u)$ for all $u \in U$.
2. $\gamma(U \times \{v\}) = \beta(v)$ for all $v \in V$.

Then $cost(\gamma) = \sum_{u \in U} \sum_{v \in V} \gamma(u, v) cost(u, v)$, 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*. The left part of Figure 5 shows the instance of the transportation problem which results from the situation in Figure 4. A solution of the problem is given by the following: Move $\frac{1}{4}$ from $\{z, x\}$ to $\{\bar{x}, x\}$, move $\frac{1}{6}$ from $\{z, y\}$ to $\{\bar{x}, y\}$, move $\frac{1}{4}$ from $\{z, \bar{z}\}$ to $\{\bar{x}, \bar{z}\}$, move $\frac{1}{4}$ from $\{z, \bar{z}\}$ to $\{\bar{x}, z\}$, each at cost 1; and $\frac{1}{12}$ from $\{z, y\}$ to $\{\bar{x}, x\}$ at cost 2. Thus the total cost of the algorithm in this step is $\frac{13}{12}$. (This number also appears as the first value under the arrow from D_{xyz} to W in Figure 4.)

Finally we fix competitiveness $C = \frac{19}{12}$. We are now ready to check the update condition 1 for this step. We have $\Phi(S) = \frac{7}{6}$; $cost_{\mathcal{A}} = \frac{13}{12}$, $C \cdot adjust = \frac{19}{12} \cdot \frac{2}{3}$ and $\sum_i \lambda_i \Phi(S_i) = \frac{10}{9}$. Thus $\Phi(S) - cost_{\mathcal{A}} + C \cdot adjust - \sum_i \lambda_i \Phi(S_i) \geq 0$.

A complete listing of moves of the algorithm is given in columns one to three of Table 1.

We have:

Theorem 1. *The knowledge state algorithm of Table 1 is C -competitive with $C = \frac{19}{12}$.*

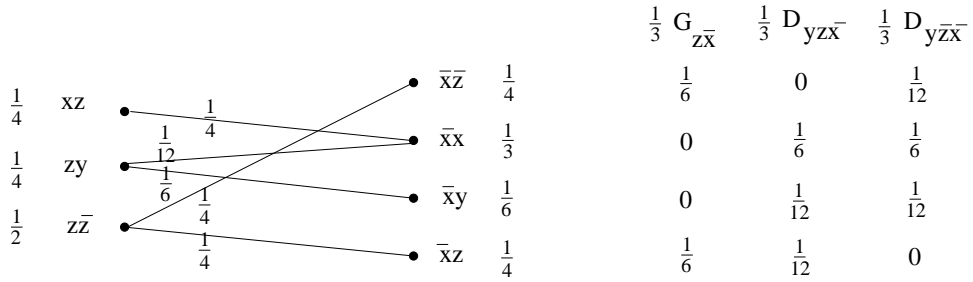


Fig. 5: The Distributional Transportation Problem

Proof. Update condition 1 is verified for every step in Table 1.

3 The Wireframe Algorithm

The knowledge state algorithm described in the previous section was analyzed in the mixed model of computation. We recall that this implies that there is a competitive behavioral online algorithm. The following lemma is well-known. (It is, for example, implicit in Chapter 6 of [7].)

Lemma 2. *The mixed model and the behavioral model of randomized online algorithms are equivalent, in the following sense. If \mathcal{A}_1 is an algorithm of one of the models, there exist an algorithm \mathcal{A}_2 of the other model, such that, given any request sequence ρ , the cost of \mathcal{A}_2 for ρ is no greater than the cost of \mathcal{A}_1 .*

We will now translate our algorithm into behavioral form, a form in which it is easy to implement the algorithm into an actual working computer program. The resulting behavioral algorithm is called the “wireframe algorithm.” At each step, in addition to the position of the two servers, the algorithm also keeps track of certain points in an octahedron. This can be best illustrated by a wireframe of an octahedron; see Figure 6. In the situation depicted in Figure 6 (see top octahedron) the server positions are at points y and z and the algorithm keeps track of constellation d_{xyz} . (See the dashed lines.) Note that constellation d_{xyz} can be best thought of as the wireframe shown in Figure 6 (top part). Next, the Figure (lower part) shows the behavior of the algorithm if \bar{x} is requested. With probability $\frac{1}{6}$ the algorithm moves the server at point y to point \bar{x} and the server at point z to x and goes into state (*i.e.* wireframe) $d_{y\bar{z}\bar{x}}$. With remaining probability $\frac{1}{6}$ the algorithm does exactly the same server movements (y to \bar{x} and z to x) and goes into state (*i.e.* wireframe) $d_{y\bar{z}\bar{x}}$. Furthermore, with equal probabilities $\frac{1}{3}$ the wireframe algorithm moves the server at point z to point \bar{x} and goes into state (*i.e.* wireframe) $d_{yz\bar{x}}$ or $d_{y\bar{z}\bar{x}}$.

Note the following:

- The algorithm has no concept of knowledge states, it merely remembers where the servers are and keeps track of only very limited extra information.

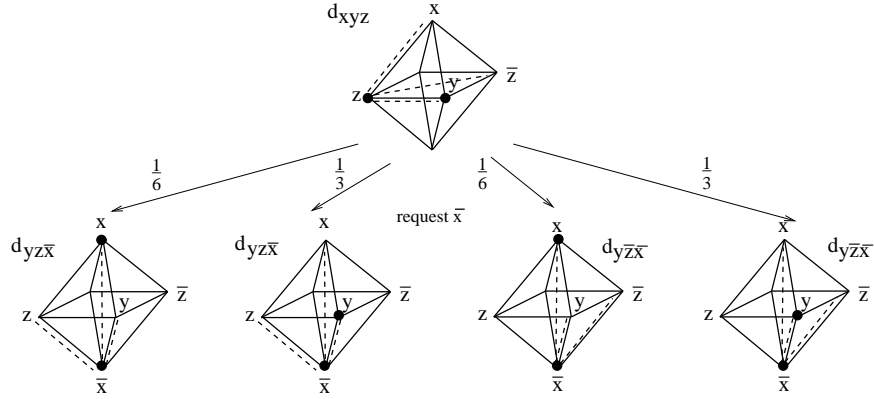


Fig. 6: One Move of the Wireframe Algorithm

Upon a request, depending on this extra information, the algorithm then decides how to move the servers and how to update its information.

- A server algorithm is called *lazy* if, in a step, it only moves one server to serve a request and it does not move any other server. Note that the algorithm is non-lazy.
- In the situation described in Figure 6 the request to \bar{x} is served by moving the server at point z to point \bar{x} with probability $\frac{2}{3} = \frac{1}{3} + \frac{1}{3}$, but the constellation memorized by the algorithm after the move is either $d_{yz\bar{x}}$ or $d_{y\bar{z}\bar{x}}$. With probability $\frac{1}{3} = \frac{1}{6} + \frac{1}{6}$, the algorithm moves the server at point y to point \bar{x} and the server at point z to x . (Again, with the caveat that two different constellations are possible after the move.)

For the behavioral move just described we will use the notation:

$$\boxed{d_{xyz}, yz \mid \bar{x} \left\{ \frac{1}{6} \{d_{yz\bar{x}}, x\bar{x}\} + \frac{1}{3} \{d_{yz\bar{x}}, y\bar{x}\} + \frac{1}{6} \{d_{y\bar{z}\bar{x}}, x\bar{x}\} + \frac{1}{3} \{d_{y\bar{z}\bar{x}}, y\bar{x}\} \right\}}$$

Using this notation, the wireframe algorithm is completely described in Table 2.

The derivation of the behavioral algorithm from the mixed algorithm is routine (it is described in general in [5, 6]); we will briefly discuss how the step of Figure 6 results from translating the transition of Figure 4. The translation uses the solution to transportation problem of Figure 5. Note that there is probability mass of $\frac{1}{4}$ at $\{(z, y)\}$. Mass $\frac{1}{12}$ is moved to $\{(\bar{x}, x)\}$ and mass $\frac{1}{6}$ is moved to $\{(\bar{x}, y)\}$. Thus, *given* that the constellation is $\{(z, y)\}$ the conditional probabilities for $\{(\bar{x}, x)\}$ and $\{(\bar{x}, y)\}$ are $\frac{1}{3}$ and $\frac{2}{3}$ respectively. Following Figure 5 to the right, we see that the mass $\frac{1}{3}$ at $\{(\bar{x}, x)\}$ is equally divided between constellations $d_{yz\bar{x}}$ and $d_{y\bar{z}\bar{x}}$. The same is true for the mass at $\{(\bar{x}, y)\}$. We conclude that the algorithm chooses with probability $\frac{1}{6}$ servers at $\{(x\bar{x})\}$ with $d_{yz\bar{x}}$, with

probability $\frac{1}{3}$ servers at $\{(y\bar{x})\}$ with $d_{yz\bar{x}}$, with probability $\frac{1}{6}$ servers at $\{(x\bar{x})\}$ with $d_{y\bar{z}\bar{x}}$, and with probability $\frac{1}{3}$ servers with $\{(y\bar{x})\}$ with $d_{y\bar{z}\bar{x}}$.

In summary we have:

Theorem 2. *The wireframe algorithm of Table 2 is C -competitive with $C = \frac{19}{12}$.*

4 The Lower Bound

Indeed, the competitiveness of the wireframe algorithm is best possible:

Theorem 3. *Let \mathcal{A} be any randomized online algorithm for the 2-server problem for $M_{2,4}$. Let C be the competitiveness of \mathcal{A} . Then $C \geq \frac{19}{12}$.*

Proof. We only give a sketch; the formal proof will be given in the full paper. We refer to Figure 7. Consider the cross-polyhedron $\{x, \bar{x}, y, \bar{y}, z, \bar{z}\}$. Without loss of generality, the initial server position is $\{(x, y)\}$. We call this situation START. Now consider:

1. Adversary requests z ; pays 1.
2. \mathcal{A} serves request; pays 1.
3. Without loss of generality, \mathcal{A} is at x with probability $\leq \frac{1}{2}$.
4. Adversary requests \bar{y} ; pays 1.
5. \mathcal{A} serves request; pays 1.

It can easily be argued that the probability that \mathcal{A} has a server at y is not smaller than $\frac{1}{2}$ and that the probability that there is a server at x is not larger than $\frac{1}{4}$.

6. Adversary requests x , pays 0.
7. \mathcal{A} serves the request; pays $\geq \frac{3}{4}$.

We call this situation the MIDDLE, see Figure 7.

Let

$$\begin{aligned} p &= \text{probability there is a server at } \bar{y} \\ q &= \text{probability there is a server at } z \\ r &= \text{probability there is a server at } y \end{aligned}$$

Then $p + q + r = 1$.

Case i: $q + 2r \geq \frac{5}{12}$ (Return to START)

- a) Adversary repeatedly requests \bar{y}, x , *i.e.* hammers at $\{(\bar{y}, x)\}$; pays 0.
- b) \mathcal{A} must move server from y and z to \bar{y} ; pays $q + 2r$.

Situation has returned to START: Adversary has paid 2, \mathcal{A} has paid no less than $2 + \frac{3}{4} + \frac{5}{12} = \frac{38}{12}$. Thus the ratio is no less than $\frac{19}{12}$.

Case ii: $q + 2r < \frac{5}{12}$ (Cycle back to MIDDLE)

References

1. Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theoret. Comput. Sci.*, 234:203–218, 2000.
2. Yair Bartal, Marek Chrobak, and Lawrence L. Larmore. A randomized algorithm for two servers on the line. In *Proc. 6th European Symp. on Algorithms (ESA)*, Lecture Notes in Comput. Sci., pages 247–258. Springer, 1998.
3. Wolfgang Bein, Lawrence L. Larmore, and John Noga. Equitable revisited. In *Proceedings of the 15th Annual European Symposium on Algorithms*, volume 4698 of *Lecture Notes in Computer Science*, pages 419–426. Springer, 2007.
4. Wolfgang Bein, Lawrence L. Larmore, and Rüdiger Reischuk. Knowledge states for the caching problem in shared memory multiprocessor systems. In *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks*, pages 307 – 312. IEEE, 2004.
5. Wolfgang Bein, Lawrence L. Larmore, and Rüdiger Reischuk. Knowledge state algorithms: Randomization with limited information. *Arxiv: archive.org/cs/0701142*, 2007.
6. Wolfgang Bein, Lawrence L. Larmore, and Rüdiger Reischuk. Knowledge states: A tool for randomized online algorithms. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (CD-ROM)*. IEEE Computer Society Press, 10 pages, 2008.
7. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
8. Marek Chrobak, Howard Karloff, Tom H. Payne, and Sundar Vishwanathan. New results on server problems. *SIAM J. Discrete Math.*, 4:172–181, 1991.
9. Marek Chrobak and Lawrence L. Larmore. An optimal online algorithm for k servers on trees. *SIAM J. Comput.*, 20:144–148, 1991.
10. Marek Chrobak, Lawrence L. Larmore, Carsten Lund, and Nick Reingold. A better lower bound on the competitive ratio of the randomized 2-server problem. *Inform. Process. Lett.*, 63:79–83, 1997.
11. Elias Koutsoupias and Christos Papadimitriou. Beyond competitive analysis. In *Proc. 35th Symp. Foundations of Computer Science (FOCS)*, pages 394–400. IEEE, 1994.
12. Elias Koutsoupias and Christos Papadimitriou. On the k -server conjecture. *J. ACM*, 42:971–983, 1995.
13. Carsten Lund and Nick Reingold. Linear programs for randomized on-line algorithms. In *Proc. 5th Symp. on Discrete Algorithms (SODA)*, pages 382–391. ACM/SIAM, 1994.
14. Mark Manasse, Lyle A. McGeoch, and Daniel Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11:208–230, 1990.
15. Ludwig Schläfli. *Theorie der vielfachen Continuität*. Birkhäuser, Basel, 1857.

a_{xz}, xz	\bar{x}	$1\{c_x, x\bar{x}\}$
a_{xz}, xz	r	$\frac{1}{2}\{b_{x zr}, xr\} + \frac{1}{2}\{b_{x zr}, zr\}$
b_{xyz}, xz	x	$1\{a_{xz}, xz\}$
b_{xyz}, yz	x	$1\{a_{xz}, xz\}$
b_{xyz}, xz	\bar{x}	$1\{d_{yz\bar{x}}, x\bar{x}\}$
b_{xyz}, yz	\bar{x}	$\frac{1}{2}\{d_{yz\bar{x}}, y\bar{x}\} + \frac{1}{2}\{d_{yz\bar{x}}, z\bar{x}\}$
b_{xyz}, xz	\bar{z}	$1\{b_{xy\bar{z}}, x\bar{x}\}$
b_{xyz}, yz	\bar{z}	$1\{b_{xy\bar{z}}, y\bar{x}\}$
b_{xyz}, xz	r	$\frac{1}{3}\{b_{x yr}, xr\} + \frac{1}{3}\{b_{x zr}, xr\} + \frac{1}{6}\{b_{x zr}, zr\} + \frac{1}{6}\{b_{y zr}, zr\}$
b_{xyz}, yz	r	$\frac{1}{3}\{b_{x yr}, yr\} + \frac{1}{6}\{b_{x zr}, zr\} + \frac{1}{3}\{b_{y zr}, yr\} + \frac{1}{6}\{b_{y zr}, zr\}$
$c_{xz}, z\bar{z}$	r	$\frac{1}{2}\{g_{zr}, zr\} + \frac{1}{2}\{g_{zr}, \bar{z}r\}$
d_{xyz}, xz	x	$1\{e_{\bar{z}x}, yz\}$
d_{xyz}, yz	x	$1\{e_{\bar{z}x}, yz\}$
$d_{xyz}, \bar{z}z$	x	$\frac{7}{12}\{e_{\bar{z}x}, yz\} + \frac{5}{12}\{e_{\bar{z}x}, y\bar{z}\}$
d_{xyz}, xz	\bar{z}	$1\{c_z, z\bar{z}\}$
d_{xyz}, yz	\bar{z}	$1\{c_z, z\bar{z}\}$
$d_{xyz}, \bar{z}z$	\bar{z}	$1\{c_z, z\bar{z}\}$
d_{xyz}, xz	r	$1\{b_{x zr}, xr\}$
d_{xyz}, yz	r	$\frac{1}{2}\{b_{x zr}, zr\} + \frac{1}{2}\{b_{y \bar{z}r}, \bar{z}r\}$
$d_{xyz}, \bar{z}z$	r	$1\{b_{y \bar{z}r}, yr\}$
d_{xyz}, xz	\bar{x}	$\frac{1}{2}\{d_{yz\bar{x}}, x\bar{x}\} + \frac{1}{2}\{d_{y\bar{z}\bar{x}}, x\bar{x}\}$
d_{xyz}, yz	\bar{x}	$\frac{1}{6}\{d_{yz\bar{x}}, x\bar{x}\} + \frac{1}{3}\{d_{y\bar{z}\bar{x}}, y\bar{x}\} + \frac{1}{6}\{d_{y\bar{z}\bar{x}}, x\bar{x}\} + \frac{1}{3}\{d_{y\bar{z}\bar{x}}, y\bar{x}\}$
$d_{xyz}, \bar{z}z$	\bar{x}	$\frac{1}{3}\{g_{z\bar{x}}, z\bar{x}\} + \frac{1}{3}\{g_{z\bar{x}}, \bar{z}\bar{x}\} + \frac{1}{6}\{d_{y\bar{z}\bar{x}}, z\bar{x}\} + \frac{1}{6}\{d_{y\bar{z}\bar{x}}, \bar{z}\bar{x}\}$
e_{xz}, xz	x	$1\{f_{zx}, zx\}$
$e_{xz}, \bar{x}z$	x	$1\{f_{zx}, \bar{x}x\}$
e_{xz}, xz	\bar{z}	$1\{a_{\bar{x}z}, \bar{x}z\}$
$e_{xz}, \bar{x}z$	\bar{z}	$1\{a_{\bar{x}z}, \bar{x}z\}$
e_{xz}, xz	r	$1\{b_{\bar{x}zr}, zr\}$
$e_{xz}, \bar{x}z$	r	$\frac{12}{19}\{b_{\bar{x}zr}, \bar{x}r\} + \frac{7}{19}\{b_{\bar{x}zr}, zr\}$
e_{xz}, xz	\bar{x}	$1\{a_{\bar{x}z}, \bar{x}z\}$
$e_{xz}, \bar{x}z$	\bar{x}	$1\{a_{\bar{x}z}, \bar{x}z\}$

f_{xz}, xz	x	$1\{e_{zx}, zx\}$
$f_{xz}, \bar{z}z$	x	$\frac{14}{19}\{e_{zx}, zx\} + \frac{5}{19}\{e_{zx}, \bar{z}x\}$
f_{xz}, xz	\bar{z}	$1\{c_z, z\bar{z}\}$
$f_{xz}, \bar{z}z$	\bar{z}	$1\{c_z, z\bar{z}\}$
f_{xz}, xz	\bar{x}	$1\{h_{z\bar{x}}, x\bar{x}\}$
$f_{xz}, \bar{z}z$	\bar{x}	$\frac{1}{19}\{h_{z\bar{x}}, x\bar{x}\} + \frac{9}{19}\{h_{z\bar{x}}, z\bar{x}\} + \frac{9}{19}\{h_{z\bar{x}}, \bar{z}\bar{x}\}$
f_{xz}, xz	r	$1\{a_{xr}, xr\}$
$f_{xz}, \bar{z}z$	r	$\frac{7}{19}\{a_{xr}, xr\} + \frac{6}{19}\{g_{zr}, zr\} + \frac{6}{19}\{g_{zr}, \bar{z}r\}$
g_{xz}, xz	x	$1\{a_{xz}, zx\}$
$g_{xz}, \bar{x}z$	x	$1\{a_{xz}, zx\}$
g_{xz}, xz	\bar{z}	$1\{c_z, z\bar{z}\}$
$g_{xz}, \bar{x}z$	\bar{z}	$1\{c_z, z\bar{z}\}$
g_{xz}, xz	r	$1\{a_{xr}, xr\}$
$g_{xz}, \bar{x}z$	r	$1\{a_{xr}, xr\}$
h_{xz}, xz	r	$\frac{2}{3}\{b_{x zr}, xr\} + \frac{1}{3}\{b_{x zr}, zr\}$
$h_{xz}, \bar{x}z$	r	$\frac{1}{3}\{b_{x zr}, zr\} + \frac{2}{3}\{b_{\bar{x}\bar{z}r}, \bar{x}r\}$
$h_{xz}, \bar{z}z$	r	$1\{b_{\bar{x}\bar{z}r}, \bar{z}r\}$
h_{xz}, xz	x	$1\{e_{\bar{z}x}, xz\}$
$h_{xz}, \bar{x}z$	x	$1\{e_{\bar{z}x}, xz\}$
$h_{xz}, \bar{z}z$	x	$\frac{1}{6}\{e_{\bar{z}x}, xz\} + \frac{5}{6}\{e_{\bar{z}x}, x\bar{z}\}$
h_{xz}, xz	\bar{z}	$1\{c_z, z\bar{z}\}$
$h_{xz}, \bar{x}z$	\bar{z}	$1\{c_z, z\bar{z}\}$
$h_{xz}, \bar{z}z$	\bar{z}	$1\{c_z, z\bar{z}\}$

Table 2: The Moves of the Wireframe Algorithm