

# A Better Algorithm for Uniform Metrical Task Systems with Few States

Wolfgang Bein and Lawrence L. Larmore\*  
 School of Computer Science  
 University of Nevada, Las Vegas  
 Las Vegas, NV 89154  
 bein@cs.unlv.edu, larmore@cs.unlv.edu

John Noga  
 Department of Computer Science  
 California State University, Northridge  
 Northridge, CA 91330  
 jnoga@csun.edu

## Abstract

We give a randomized algorithm (the "Wedge Algorithm") of competitiveness  $\frac{3}{2}H_k - \frac{1}{2k}$  for any metrical task system on a uniform space of  $k$  points, for any  $k \geq 2$ , where  $H_k = \sum_{i=1}^k \frac{1}{i}$ , the  $k^{\text{th}}$  harmonic number. This algorithm has better competitiveness than the Irani-Seiden algorithm if  $k$  is smaller than  $10^8$ . The algorithm is better by a factor of 2 if  $k < 47$ .

Keywords: *Online Algorithms; Randomized Algorithms; Task Systems.*

## 1 The Ice Cream Problem and Other Preliminaries

We reprise a simple problem taken from [7], the "ice cream problem." Suppose that there is a machine that makes two flavors of ice cream, vanilla and chocolate. The machine has two states,  $V$  and  $C$ . In state  $V$ , the machine can make vanilla ice cream at a cost of \$1 per gallon. In state  $C$ , the machine can make chocolate ice cream at a cost of \$2 per gallon. At a cost of \$1, the state of the machine can be changed. The ice cream vendor also has the option of making the ice cream by hand, not using the machine, at a cost of \$2 per gallon for vanilla and \$4 per gallon for chocolate. When a customer arrives and requests a gallon of ice cream of a given flavor, the vendor can either use the machine, changing its state if necessary, or make the ice cream by hand.

The ice cream problem is a simple metrical task system. The states are  $V$  and  $C$ , and the distance between them is 1. The requests are  $v$ , request vanilla, and  $c$ ,

request chocolate. Each task is a function on the states, as given by Table 1.

	$V$	$C$
$v$	1	2
$c$	4	2

**Table 1. State table for the ice cream problem**

More generally, a metrical task system is given by a set of states drawn from a metric space  $\mathcal{M}$ , as well as a set of tasks  $\mathcal{T}$ , where a task is a function  $\tau: \mathcal{M} \rightarrow \mathbb{R}^+$ . We concentrate on the case of a uniform metrical task system with  $k$  states  $A_1 \dots A_k$ ; in this case we have  $\|A_j, A_i\| = 1$ , if  $i \neq j$ . Furthermore, for convenience, we often write  $\tau = (\tau(A_1), \dots, \tau(A_k))$ , or simply  $\tau = (\tau_1, \tau_2, \dots, \tau_k)$ .

Intuitively, a task can be "served" at cost  $\tau(A_i)$  when the system is in state  $A_i \in \mathcal{M}$ . We consider the online problem, where a task request sequence  $\varrho = \tau^1, \tau^2 \dots \tau^m$  is given, and a task must be served without knowledge of future requests<sup>1</sup>. Assuming the system be in some state  $A_{i_{t-1}}$  at the beginning of step  $t < m$ , an online algorithm  $\mathcal{A}$  can move to any other state  $A_{i_t}$ ,  $i_t \neq i_{t-1}$  at cost 1 or remain in the current state at cost 0. The cost of transitioning to the new state  $A_{i_t}$  is called the *movement cost*. The cost  $\tau(A_{i_t})$  to serve the request from the chosen state is called *service cost*. Thus, given initial state  $A_{i_0}$  and choice of states  $\bar{A} = A_{i_1}, \dots, A_{i_m}$ , the cost to service  $\varrho$ , is  $\text{cost}(A_{i_0}, \bar{A}, \varrho) = \sum_{t=1}^m (\|A_{i_{t-1}}, A_{i_t}\| + \tau(A_{i_t}))$ .

As is customary, we analyze an online algorithm for metrical system in terms of its *competitive ratio*, which

\*Research of these authors supported by NSF grant CCR-0312093.

<sup>1</sup>We remind the reader that we use subscripts to indicate components and superscripts to indicate time. Thus,  $\tau_i$  is the  $i^{\text{th}}$  component of request  $\tau$ , while  $\tau^t$  is the request at step  $t$ .

essentially gives the ratio of its cost over the cost of an optimal algorithm which has knowledge of the entire request sequence before making any decisions. More precisely, we say that an online algorithm  $\mathcal{A}$  for a given metrical task system is  $C$ -competitive if there is a constant  $K$  such that, given any request sequence  $\rho$ , and algorithm  $\mathcal{A}$ 's choice of states,  $\bar{A}$ ,

$$\text{cost}(A_{i_0}, \bar{A}, \rho) \leq C_{\text{OPT}}(A_{i_0}, \rho) + K$$

where

$$C_{\text{OPT}}(A_{i_0}, \rho) = \min_{\bar{A}} \{ \text{cost}(A_{i_0}, \bar{A}, \rho) \}.$$

There is a  $\frac{7}{6}$ -competitive online algorithm for the ice cream problem given in [7], as follows:

- If the state is  $V$  and the request is  $v$ , or if the state is  $C$  and the request is  $c$ , then simply use the machine.
- If the state is  $V$  and the request is  $c$ , then change the state of the machine to  $C$  and use the machine.
- If the state is  $C$  and the request is  $v$ , and if the previous request was also  $v$ , then change the state of the machine to  $V$  and use the machine. Otherwise, make the ice cream by hand.

It is also shown in [7] that no deterministic online algorithm for the ice cream problem can have competitiveness less than  $\frac{7}{6}$ .

For the general deterministic problem, Borodin, Linial, and Saks [4, 5] show that any metrical task system with  $n$  states has a  $(2n - 1)$ -competitive online algorithm. Furthermore, this result is best possible in general, *i.e.*, there is no lower competitiveness which applies to all  $n$ -state task systems. However, much better competitiveness can be achieved using randomization.

Throughout this paper we describe randomized algorithms in distributional form, *i.e.*, at each step the algorithm is specified by a distribution  $\pi = (p_1, \dots, p_k)$  over its states  $A_1 \dots A_k$ , and the corresponding service and movement costs are defined in terms of expected value. Thus, for randomized algorithms, competitiveness is stated in terms as expected value as well.

There is no randomized online algorithm for the ice cream problem of competitiveness less than  $\frac{11}{10}$ . We give an  $\frac{11}{10}$ -competitive randomized online algorithm, which we describe using the distribution model. Let  $\mathcal{D} = \{V, \frac{2}{5}V + \frac{3}{5}C, C\}$ , a set of distributions on the set of states. The initial distribution is some  $\pi^0 \in \mathcal{D}$ . After  $t$  steps, the current distribution will be some  $\pi^t \in \mathcal{D}$ .

If the request at step  $t$  is  $\tau^t \in \{v, c\}$ , then  $\pi^t$  is computed as a function of  $\pi^{t-1}$  and  $\tau^t$ , as shown by Table 2. The algorithm thus remembers nothing except its current distribution.

<b>if <math>\pi^{t-1}</math> is:</b>	$V$	$\frac{2}{5}V + \frac{3}{5}C$	$C$
<b>then for <math>v</math> go to <math>\pi^t</math>:</b>	$V$	$V$	$\frac{2}{5}V + \frac{3}{5}C$
<b>then for <math>c</math> go to <math>\pi^t</math>:</b>	$C$	$C$	$C$

**Table 2. Transitions of the ice cream algorithm**

The  $\frac{11}{10}$ -competitiveness of this algorithm can be verified by using the potential

$$\begin{aligned} \Phi(V) &= 0 \\ \Phi(\frac{2}{5}V + \frac{3}{5}C) &= \frac{1}{2} \\ \Phi(C) &= \frac{3}{10} \end{aligned}$$

and the methods given in Section 2.

We note that task systems have been of significant interest (see, *e.g.* [1, 2, 5, 6, 7, 9, 10]) because many other online optimization problems can be modeled as task system problems.

In the next section we give a new randomized algorithm for any metrical task system on a uniform space of  $k$  points, for any  $k \geq 2$ . We call this algorithm the ‘‘Wedge Algorithm.’’ The best known algorithm for this problem is the algorithm given by Irani and Seiden in [9] which has competitive ratio  $H_k + O(\sqrt{\log k})$ . ( $H_k = \sum_{i=1}^k \frac{1}{i}$  is the  $k^{\text{th}}$  harmonic number.) This algorithm is asymptotically optimal but quite far from optimal if the number of states is not very large. For example, the competitiveness of the Wedge Algorithm for  $k = 3$  is  $\frac{31}{12} \approx 2.58$ , whereas the Irani-Seiden algorithm has competitiveness approximately 9.18. Table 3 shows competitive ratios for some other  $k$ . We will show that the competitive ratio of the Wedge Algorithm is  $\frac{3}{2}H_k - \frac{1}{2k}$  and we note that the competitive ratio of the Irani-Seiden Algorithm is about double the ratio of the Wedge Algorithm if the number of states is less than 47.

An important concept in the definition of the algorithm is the concept of a work function, which we briefly review here. (See [3] for a more extensive discussion of work functions and their properties.) Assume  $A_1$  to be a start state. Let  $\rho = \tau^1, \dots, \tau^n$  be given to be the sequence of tasks. For any  $t \leq n$ , let  $\omega_\rho^t(A_i)$  be the minimum cost of servicing the sequence  $\tau^1, \dots, \tau^t$  starting at state  $A_1$  and ending at state  $A_i$ . Thus,  $\omega_\rho^t$

k	Wedge	Irani-Seiden
3	2.58333	9.18338
7	3.81786	10.4427
10	4.34345	10.9829
10 <sup>2</sup>	7.77607	14.4312
10 <sup>3</sup>	11.2277	17.7313
10 <sup>8</sup>	25.043	29.9873

**Table 3. Competitive ratios of the Irani-Seiden and the Wedge Algorithm given different numbers of states**

is a real valued function on  $M$ , which we call the *work function* at step  $t$ . If  $\varrho$  is understood, we simply write  $\omega^t$  for  $\omega_\varrho^t$ . For convenience, we write any work function  $\omega$  as a  $k$ -tuple  $(x_1, \dots, x_k)$ , where  $x_i = \omega(A_i)$ .

From [3], we have the following in the case of the uniform task system with  $k$  states:

- [Initial condition]  $\omega^0(A_1) = 0$  and  $\omega^0(A_i) = \|A_1, A_i\| = 1$  for all  $i \neq 1$ .
- [Optimality]  $\min_i \{\omega^n(A_i)\}$  is the minimum cost of servicing the sequence  $\varrho$ .
- [Lipschitz]  $|\omega^t(A_i) - \omega^t(A_j)| \leq \|A_i, A_j\| = 1$  for any  $t$  and any  $1 \leq i, j \leq k$ .
- [Update operator] If  $\omega^{t-1} = (x_1, \dots, x_k)$  and  $\tau^t = (c_1, \dots, c_k)$ , then  $\omega^t = (y_1, \dots, y_k) = \omega^{t-1} \wedge \tau^t$ , defined by

$$y_i = \min \left\{ \begin{array}{l} x_i + c_i \\ \min_{j \neq i} \{x_j + c_j + 1\} \end{array} \right\}$$

## 2 The Wedge Algorithm

The Wedge Algorithm, which we refer to as  $\mathcal{A}_k$ , is a *stable distribution* algorithm, *i.e.*, its distribution depends only on the current work function. Let  $\mathcal{W}$  be the set of all non-negative functions  $M \rightarrow \mathfrak{R}^k$  which satisfy the Lipschitz condition, and  $\Pi$  the set of all possible distributions on  $M$ . A stable distribution algorithm  $\mathcal{A}$  is defined by a function  $\pi_{\mathcal{A}} : \mathcal{W} \rightarrow \Pi$ , as follows: if  $\omega^t$  is the work function after  $t$  steps, then the distribution of states of  $\mathcal{A}$  after  $t$  steps is  $\pi_{\mathcal{A}}(\omega^t)$ .

In our example, where there are  $k$  states, any distribution on  $M$  can be written as a  $k$ -tuple of non-negative numbers whose sum is 1. Thus

$$\pi_{\mathcal{A}_k} : \mathcal{W} \subseteq \mathfrak{R}^k \rightarrow \Pi \subseteq \mathfrak{R}^k$$

Before we give the general definition of  $\mathcal{A}_k$ , we summarize Chrobak and Noga's definition of  $\mathcal{A}_2$  and  $\mathcal{A}_3$

[8]. They define  $\pi_{\mathcal{A}_2}$  as follows:

$$\pi_{\mathcal{A}_2}(x_1, x_2) = \left( \frac{x_2 - x_1 + 1}{2}, \frac{x_1 - x_2 + 1}{2} \right)$$

Note that, for any  $x$ ,  $\pi(x, x+1) = (1, 0)$ ,  $\pi(x, x) = (\frac{1}{2}, \frac{1}{2})$ , and  $\pi(x+1, x) = (0, 1)$ .

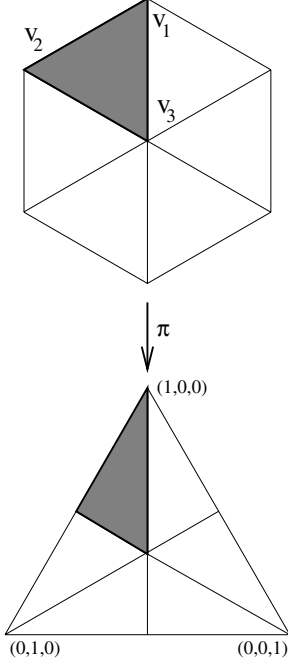
We define  $\pi = \pi_{\mathcal{A}_3}$  by using the symmetry of  $M$  under any permutation of the three states:  $\mathcal{W}$  is the union of six *regions* characterized by the ordering of the components of the work function. Let  $\Delta$  be the set of all work functions  $\omega = (x_1, x_2, x_3)$  such that  $x_1 \leq x_2 \leq x_3 \leq x_1 + 1$ . We call  $\Delta$  the *model region*. If  $\sigma$  is any permutation of the three states (*i.e.*, of the three components of  $\omega$ ), then  $\sigma\Delta$  is a region.  $\Delta$  is three dimensional, but we adopt the *offset* rule that  $\pi(x_1, x_2, x_3) = \pi(x_1 + h, x_2 + h, x_3 + h)$ , so it suffices to define  $\pi$  on a 2-dimensional cross-section of  $\Delta$ . We identify three *vertices* of  $\Delta$ , namely  $v_1 = (0, 1, 1)$ ,  $v_2 = (0, 0, 1)$ , and  $v_3 = (0, 0, 0)$ . Let

- $\pi_{\mathcal{A}_3}(v_1) = (1, 0, 0)$ .
- $\pi_{\mathcal{A}_3}(v_2) = (\frac{1}{2}, \frac{1}{2}, 0)$ .
- $\pi_{\mathcal{A}_3}(v_3) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ .

We then define  $\pi_{\mathcal{A}_3} : \mathcal{W} \rightarrow \mathfrak{R}$  to be the unique extension which is linear on  $\Delta$ , follows the offset rule, and is symmetric under permutation of the states. The reader should note that  $\pi_{\mathcal{A}_3}$  is continuous on  $\mathcal{W}$  and linear on each region, but not linear on  $\mathcal{W}$ . Figure 1 indicates the function  $\pi_{\mathcal{A}_3}$  as a mapping from a cross-section of  $\mathcal{W}$ , a hexagon, to  $\Pi$ , a triangle. The model region is shaded in the above figure, while the image of the model region is shaded in the lower figure.

Now, for any  $k$ , let  $\Delta \subseteq \mathfrak{R}^k$ , the model region, be the set of work functions  $\omega = (x_1, x_2, \dots, x_k)$  such that  $x_1 \geq 0$ ,  $x_i \leq x_{i+1}$  for all  $1 \leq i < k$ , and  $x_k \leq x_1 + 1$ . For  $1 \leq m \leq k$ , let  $v_m$  be the  $m^{\text{th}}$  *vertex* of  $\Delta$ , defined to be that work function which is 0 on the first  $m$  coordinates and 1 on all others. We then define  $\pi_{\mathcal{A}_k}(v_m)$  to be the distribution which is  $\frac{1}{m}$  on each of  $A_1, \dots, A_m$  and zero on other states. We then extend  $\pi_{\mathcal{A}_k}$  to all of  $\mathcal{W}$  by requiring that it satisfy the offset rule, that it be linear on  $\Delta$ , and that it be symmetric under permutation of the states. Note that  $\pi_{\mathcal{A}_k}$  is continuous on  $\mathcal{W}$ , is linear on each of the  $k!$  regions, but is not linear on  $\mathcal{W}$  for  $k > 2$ .

We now analyze the competitiveness of the  $\mathcal{A}_k$ . If  $\pi = (p_1, \dots, p_k) \in \Pi$  and  $\tau = (\tau_1, \dots, \tau_k)$  is a task, define  $\text{cost}(\pi, \tau) = \sum_{i=1}^k p_i \tau_i$ , the cost of executing the task while at the distribution. The following lemma is taken from Section 1.4 of [3].



**Figure 1. The Distribution Function  $\pi$**

**Lemma 1** *If there is a function  $\Phi : \mathcal{W} \rightarrow \mathfrak{R}$  which satisfies the following conditions for some  $C > 0$ :*

- (a) *If  $\omega$  is any work function and  $h \geq 0$  is a constant,  $\Phi(\omega + h) = \Phi(\omega) - C \cdot h$ .*
- (b) *If  $\omega$  is any work function and let  $\tau$  be any task, then  $\Phi(\omega) \geq \Phi(\omega \wedge \tau) + \text{cost}(\pi(\omega \wedge \tau), \tau) + \|\pi(\omega), \pi(\omega \wedge \tau)\|$ , where  $\|\pi, \pi'\|$  denotes the transportation distance between  $\pi$  and  $\pi'$ .*

then  $\mathcal{A}$  is  $C$ -competitive.

We call a function  $\Phi$  which satisfies the hypotheses of Lemma 1 a *potential* function. Note that  $\Phi$  need not be non-negative.

We now prove that  $\mathcal{A}_k$  is  $(\frac{3}{2}H_k - \frac{1}{2k})$ -competitive by defining a potential function  $\Phi$ .

Given a work function  $\omega = (x_1, \dots, x_k) \in \Delta$ , define

$$\begin{aligned} \ell_i = \ell_i(\omega) &= \begin{cases} x_{i+1} - x_i & \text{if } i < k \\ x_i + 1 - x_k & \text{if } i = k \end{cases} \\ \Phi(\omega) &= \frac{1}{2k}x_1 - \frac{3}{2}\sum_{i=1}^k \frac{1}{i}x_i - \frac{1}{4} \\ &\quad + \frac{1}{4}\sum_{1 \leq i \leq k} \ell_i^2 + \frac{1}{2}\sum_{1 \leq i < j \leq k} \frac{i}{j}\ell_i\ell_j \\ &= \frac{1}{2k}x_1 - \frac{3}{2}\sum_{i=1}^k \frac{1}{i}x_i \end{aligned}$$

$$- \frac{1}{2}\sum_{1 \leq i < j \leq k} \frac{j-i}{j}\ell_i\ell_j$$

The potential extends to all work function by symmetry. If  $\omega \notin \Delta$ , find a permutation  $\sigma$  such that  $\omega \circ \sigma \in \Delta$ . Then define  $\Phi(\omega) = \Phi(\omega \circ \sigma)$ .

**Lemma 2** *Suppose  $\omega = (x_1, \dots, x_k) \in \Delta$ . Then*

- (a) *Write  $\pi_{\mathcal{A}_k}(\omega) = (p_1, p_2, \dots, p_k)$ . Then  $p_i = p_i(\omega) = \sum_{j=i}^k \frac{1}{j}\ell_j$  for  $1 \leq i \leq k$ .*
- (b)  *$\Phi(\omega + h) = \Phi(\omega) - C \cdot h$  for  $C = \frac{3}{2}H_k - \frac{1}{2k}$  and  $h \in \mathfrak{R}$ .*
- (c)  *$\frac{\partial \Phi}{\partial x_i} + p_i - \frac{\partial p_i}{\partial x_i} \leq 0$  for  $1 \leq i \leq k$ .*

*Proof:* Part (a): Since  $\pi_i(\omega)$  is linear, we only need to verify the equality at the vertices of  $\Delta$ . For any given  $1 \leq m \leq k$ , Then  $\ell_m(v_m) = 1$ , and  $\ell_i(v_m) = 0$  for  $i \neq m$ . If  $i > m$ , then all the terms in the summation for  $p_i$  are zero, while if  $i \leq m$ , one term is  $\frac{1}{m}$  and the other terms are zero.

Part (b): Recall that  $\ell_i(\omega + h) = \ell_i(\omega)$ . Thus

$$\Phi(\omega) - \Phi(\omega + h) = -\frac{1}{2k}h + \frac{3}{2}\sum_{i=1}^k \frac{1}{i}h = \frac{3}{2}H_k - \frac{1}{2k}$$

Part (c):

If  $i = 1$ , we have

$$\begin{aligned} \frac{\partial \Phi}{\partial x_1} &+ p_1 - \frac{\partial p_1}{\partial x_1} \\ &= \frac{1}{2k} - \frac{3}{2} + \frac{1}{2}\sum_{1 < j \leq k} \frac{j-i}{j}\ell_j \\ &\quad - \frac{1}{2}\sum_{1 \leq j < k} \frac{k-j}{k}\ell_j + \sum_{j=1}^k \frac{1}{j}\ell_j \\ &\quad + 1 - \frac{1}{k} \\ &= -\left(\frac{k+1}{2k}\right)(1 - \ell_1 - \ell_k) \\ &\quad + \sum_{j=2}^{k-1} \frac{(j-1)k - (k-j)j + 2k}{2jk}\ell_j \\ &= -\left(\frac{k+1}{2k}\right)\left(1 - \sum_{j=1}^k \ell_j\right) \\ &\quad + \sum_{j=2}^{k-1} \frac{(j-1)k - (k-j)j + 2k - j(k+1)}{2jk}\ell_j \\ &= 0 + \sum_{j=2}^{k-1} \frac{-jk + j^2 + k - j}{2jk}\ell_j \end{aligned}$$

$$\begin{aligned}
&= -\sum_{j=2}^{k-1} \frac{(k-j)(j-1)}{2jk} \ell_j \\
&\leq 0
\end{aligned}$$

If  $1 < i \leq k$ , we have

$$\begin{aligned}
\frac{\partial \Phi}{\partial x_i} &+ p_i - \frac{\partial p_i}{\partial x_i} \\
&= -\frac{3}{2i} + \frac{1}{2} \sum_{i < j \leq k} \frac{j-i}{j} \ell_j \\
&\quad + \frac{1}{2} \sum_{1 \leq j \leq i-1} \frac{i-j}{i} \ell_j - \frac{1}{2} \sum_{i \leq j \leq k} \frac{j-(i-1)}{j} \ell_j \\
&\quad - \frac{1}{2} \sum_{1 \leq j < i-1} \frac{i-1-j}{i-1} \ell_j + \sum_{j=i}^k \frac{1}{j} \ell_j + \frac{1}{i} \\
&= -\frac{1}{2i}(1 - \ell_{i-1} - \ell_i) \\
&\quad + \frac{1}{2} \sum_{1 \leq j \leq i-1} \frac{(i-j)(i-1) - i(i-1-j)}{i(i-1)} \ell_j \\
&\quad + \frac{1}{2} \sum_{i < j \leq k} \frac{1}{j} \ell_j \\
&= -\frac{1}{2i} \left(1 - \sum_{j=1}^k \ell_j\right) + \sum_{1 \leq j \leq i-1} \frac{j-(i-1)}{2i(i-1)} \ell_j \\
&\quad + \sum_{i < j \leq k} \frac{i-j}{2ij} \ell_j \\
&= 0 - \sum_{1 \leq j \leq i-1} \frac{i-1-j}{2i(i-1)} \ell_j - \sum_{i < j \leq k} \frac{j-i}{2ij} \ell_j \\
&\leq 0
\end{aligned}$$

This completes the proof of (c).  $\square$

**Theorem 1** *There is a  $(\frac{3}{2}H_k - \frac{1}{2k})$ -competitive randomized online algorithm for the uniform metrical task system with  $k$  states.*

*Proof:* We need only show that  $\Phi$  satisfies the conditions of Lemma 1. condition (a) of Lemma 1 follows from Part (b) of Lemma 2.

We now prove that condition (b) of Lemma 1 holds. Let  $\omega = (x_1, \dots, x_k)$  and let  $\tau = (\tau_1, \dots, \tau_k)$ . Without loss of generality,  $\omega \in \Delta$ . Using the techniques of [3], we can assume that, for some  $m \in \{1, \dots, k\}$ ,  $\tau_m = \epsilon > 0$ , and  $\tau_i = 0$  for all  $i \neq m$ . We can also assume that one of the following two conditions holds:

- $\omega \wedge \tau = \omega$
- For some  $m$ ,  $\tau_m = \epsilon > 0$ ,  $\tau_i = 0$  for all  $i \neq m$ , and  $\omega \wedge \tau = \omega + \tau \in \Delta$ .

We first consider the case that  $\omega \wedge \tau = \omega$ . Then  $x_m = x_1 + 1$ , which implies that  $p_m(\omega) = p_m(\omega \wedge \tau) = 0$ , and the result holds trivially.

Otherwise, for any  $0 \leq t \leq \epsilon$ , let  $\tau(t)$  be the request whose  $m^{\text{th}}$  coordinate is  $t$ , and all of whose other coordinates are zero. Thus,  $\tau(0)$  is the null task, and  $\tau(\epsilon) = \tau$ . For any  $i$ , let  $x_i(t)$ ,  $\ell_i(t)$ ,  $\pi(t)$ , and  $p_i(t)$  be the values of  $x_i$ ,  $\ell_i$ ,  $\pi$ , and  $p_i$  after the request  $\omega, \tau(t)$ .  $\omega \wedge \tau(t)$  lies in the interior of  $\Delta$  for all  $0 < t < \epsilon$ . Thus,  $x_i(t)$ ,  $\ell_i(t)$ , and  $p_i(t)$  are smooth functions in the range  $0 \leq t \leq \epsilon$ . It can be easily verified that  $p_m(t)$  is monotone decreasing.

In the computation below, the first inequality is obtained by integrating the inequality given in Part (c) of Lemma 2. The second inequality follows from the monotonicity of  $p_m(t)$ .

$$\begin{aligned}
\Phi(\omega) &\geq \Phi(\omega \wedge \tau) + \int_0^\epsilon p_m(t) dt - p_m(\epsilon) + p_m(0) \\
&\geq \Phi(\omega \wedge \tau) + \epsilon p_m(\epsilon) - p_m(\epsilon) + p_m(0) \\
&= \Phi(\omega \wedge \tau) + \text{cost}(\omega \wedge \tau, \pi(\omega \wedge \tau)) \\
&\quad + \|\pi(\omega), \pi(\omega \wedge \tau)\|
\end{aligned}$$

completing thus the verification of Hypothesis (b) of Lemma 1. The result follows.  $\square$

### 3 Further Refinements

It would be desirable to know the *optimal* competitive ratio for any metrical task system on a uniform space of  $k$  points, for any fixed value of  $k$ . In this section we describe ongoing work in that direction.

The idea is that the wedges of Figure 1 could be further subdivided to obtain algorithms with better competitive ratio. We will briefly describe algorithms  $\text{SubWedge}_{k,n}$ , where  $k \geq 3$  and  $n \geq 1$ . For any  $1 \leq i < j \leq k$  and for any integer  $0 \leq m \leq n$ , let  $h_{i,j,m}$  be the hyperplane  $x_j - x_i = \frac{m}{n}$ . Writing each work function as a  $k$ -tuple  $(x_1, \dots, x_k)$ , the  $\{h_{i,j,m}\}$  subdivide  $\Delta \subseteq \mathcal{W}$  into  $n^{k-1}$  subregions. Since we will use the offset rule, we focus our attention entirely on a cross section of  $\Delta$ , which is a  $(k-1)$ -simplex. The induced subdivision of any cross-section consists of  $n^{k-1}$   $(k-1)$ -simplices, and has  $\binom{n+k-1}{k-1}$  vertices.  $\text{SubWedge}_{k,n}$  is constructed as follows:

1. Define  $\pi(v)$  for every work function  $v$  which is a vertex of the cross-section.
2. Extend  $\pi$  by the offset rule.
3. Extend  $\pi$  to all of  $\Delta$  such that  $\pi$  is linear on every subregion.

4. Extend  $\pi$  by symmetry, *i.e.*,  $\pi(\sigma(\omega)) = \sigma(\pi(\omega))$  for every permutation of the  $k$  states and every  $\omega \in \Delta$ .
5. If the current work function is  $\omega$ , the distribution of  $\text{SubWedge}_{k,n}$  is  $\pi(\omega)$ .

Notice that step 1 is left open. Thus, as it currently stands,  $\text{SubWedge}_{k,n}$  is a family of algorithms, even for a fixed  $k$  and  $n$ . If  $n = 1$ ,  $\pi(v)$  is defined as for  $\mathcal{A}_k$ . For larger  $n$ , we would like the choices in step 1 to cause the competitiveness to be minimized.

Our approach can also be used to obtain lower bounds. In fact, as the subdivision is further refined, better bounds are obtained. Table 4 shows lower bounds for  $k = 3$ .

$n$	$C_k^n$
1	1.83333
2	2.16579
3	2.28480
...	...
145	2.52721

**Table 4. Lower bounds for  $k = 3$**

Our preliminary work leads us to conjecture that for the three-state uniform metrical task system problem the true competitiveness must be approximately 2.53. It would be interesting to generalize this approach for arbitrary values of  $k$  to obtain algorithms which have competitiveness arbitrarily close to optimal. We conjecture that for appropriate choices in step 1 for any fixed  $k$ , the limiting competitiveness of  $\text{SubWedge}_{k,n}$  as  $n$  grows is the optimal randomized competitiveness of the uniform metrical task system with  $k$  states.

## References

- [1] Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog( $n$ )-competitive algorithm for metrical task systems. In *Proc. 29th Symp. Theory of Computing (STOC)*, pages 711–719. ACM, 1997.
- [2] Avrim Blum and C. Burch. On-line learning and the metrical task system problem. In *Proc. 10th Conf. on Computational Learning Theory (COLT)*, pages 45–53. ACM, 1997.
- [3] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

- [4] Allan Borodin, Nathan Linial, and Michael Saks. An optimal online algorithm for metrical task systems. In *Proc. 19th Symp. Theory of Computing (STOC)*, pages 373–382. ACM, 1987.
- [5] Allan Borodin, Nathan Linial, and Michael Saks. An optimal online algorithm for metrical task system. *Journal of the ACM*, 39:745–763, 1992.
- [6] William R. Burley and Sandy Irani. On algorithm design for metrical task systems. In *Proc. 6th Symp. on Discrete Algorithms (SODA)*, pages 420–429. ACM/SIAM, 1995.
- [7] Marek Chrobak and Lawrence L. Larmore. Metrical task systems, the server problem, and the work function algorithm. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 74–94. Springer, 1998.
- [8] Marek Chrobak and John Noga. Unpublished Manuscript, 1996.
- [9] Sandy Irani and Steven S. Seiden. Randomized algorithms for metrical task systems. *Theoretical Computer Science*, 194:163–182, 1998.
- [10] Steve S. Seiden. Unfair problems and randomized algorithms for metrical task systems. *Information and Computation*, 148:219–240, 1999.