How to Whack Moles [1]

Sandra Gutiérrez [a,2] Sven O. Krumke [b,3] Nicole Megow [d]
Tjark Vredeveld [c]

[a] *Escuela Politécnica Nacional, Facultad de Matemáticas, Quito, Ecuador. Email:* sgutier@server.epn.edu.ec

[b] *University of Kaiserslautern, Department of Mathematics, P.O. Box 3049, 67653 Kaiserslautern, Germany. Email:* krumke@mathematik.uni-kl.de

[c] *Konrad-Zuse-Zentrum für Informationstechnik Berlin, Department Optimization, Takustr. 7, 14195 Berlin-Dahlem, Germany. Email:* vredeveld@zib.de

[d] *Technische Universität Berlin, Strasse des 17. Juni 136, 10623 Berlin, Germany. Email:* nmegow@math.tu-berlin.de

In the classical *whack-a-mole game* moles that pop up at certain locations must be whacked by means of a hammer before they go under ground again. The goal is to maximize the number of moles caught. This problem can be formulated as an online optimization problem: Requests (moles) appear over time at points in a metric space and must be served (whacked) by a server (hammer) before their deadlines (i.e., before they disappear). An online algorithm learns each request only at its release time and must base its decisions on incomplete information. We study the online whack-a-mole problem (WHAM) on the real line and on the uniform metric space. While on the line no deterministic algorithm can achieve a constant competitive ratio, we provide competitive algorithms for the uniform metric space. Our online investigations are complemented by complexity results for the offline problem.

Key words: Online algorithms, competitive analysis, NP-hardness, dynamic programming

[1] Research supported by the DFG Research Center "Mathematics for key technologies" (FZT 86) in Berlin.
[2] Supported by the German Academic Exchange Service (DAAD)
[3] Supported by the German Science Foundation (DFG, grant Gr 883/10)

1 Introduction

In the popular *whack-a-mole game* moles pop up at certain holes from under ground and, after some time, disappear again. The player is equipped with a hammer and her goal is to hit as many moles as possible while they are above the ground. Clearly, from the viewpoint of the player this game is an online optimization problem since the times and positions where moles will peek out are not known in advance. She has to decide without knowledge of the future which of the currently visible moles to whack and how to move the hammer into a "promising" position. What is a good strategy for whacking moles (if there exists any)? How much better could one perform if one had magical powers and knew in advance where moles will show up? How hard is it to compute an optimal solution offline? In this paper we investigate all of the above questions. It will turn out that for the analysis two parameters play a crucial role: the time a mole stays above ground and the maximum number of moles that can be in one hole simultaneously.

The *whack-a-mole problem* with popup duration $T \geq 0$ and mole-per-hole limit N (briefly $\text{WHAM}_{T,N}$) can be formulated in mathematical terms as follows: We are given a metric space $M = (X, d)$ with a distinguished origin $0 \in X$ and a sequence $\sigma = (r_1, \ldots, r_m)$ of requests (moles). A server (hammer) moves on the metric space M at unit speed. It starts in the origin at time 0. Each request $r_j = (t_j, p_j)$ specifies a *release time* t_j and a point (hole) $p_j \in X$ where the mole pops up. The mole-per-hole-limit N denotes the maximum number of moles that are allowed to peek out of the same hole simultaneously for a positive amount of time (this includes moles that have been whacked and in reality are no longer present). A request r_j is served if the server reaches the point p_j in the time interval $[t_j, t_j + T]$. We will refer to $t_j + T$ as the *deadline* of the request. The goal is to whack as many moles as possible. If no confusion can occur we write only WHAM instead of $\text{WHAM}_{T,N}$.

An online algorithm learns about the existence of a request only at its release time. We evaluate the quality of online algorithms by *competitive analysis* [5], which has become a standard yardstick to measure the performance. An algorithm ALG for WHAM is called *c-competitive* if for any instance the number of moles whacked by ALG is at least $1/c$ times the number of moles caught by an optimal offline algorithm OPT. If ALG is randomized, then $\text{ALG}(\sigma)$ is replaced by its expected value $E[\text{ALG}(\sigma)]$ (this corresponds to the oblivious adversary model, see [5]). The *competitive ratio* of ALG is the infimum over all c such that ALG is c-competitive. In the literature, competitiveness is sometimes defined allowing an additional additive constant, b, which means for WHAM that an online algorithm may whack at most b moles less than $1/c$ times the optimum. For all our lower bounds we can easily extend the constructions (by "smart" repetition which makes the offline optimum profit arbitrarily large)

and therefore neglect this additive constant. The first two of the questions raised above amount to asking which competitive ratios are achievable by online algorithms.

In this paper we mainly study the WHAM on two different metric spaces: the (truncated) line and the uniform metric space. The (truncated) line is the classical setup for the *whack-a-mole game*. The motivation for studying the uniform metric space (a complete graph with unit edge weights) is that "in practice" it does barely matter between which points the hammer is moved: the main issue is whether the hammer is moved (and to which point) or whether it remains at its current location.

1.1 Related work.

The WHAM falls into the class of *online dial-a-ride problems*. In an online dial-a-ride problem objects must be transported between points in a metric space by a server of limited capacity. Transportation requests arrive online, specifying the objects to be transported and the corresponding source and destination. If for each request its source and destination coincide, the resulting problem is usually referred to as the online traveling salesman problem (OLTSP). The WHAM is the OLTSP with the objective to maximize the number of requests served before their deadlines.

The OLTSP has been studied for the objectives of minimizing the makespan [1,2,6], the weighted sum of completion times [6,12], and the maximum/average flow time [9,13]. Since dial-a-ride problems (where sources and destinations need not coincide) can be viewed as generalizations of scheduling problems (see e.g. [1]), lower bounds for scheduling problems carry over. In [3], Baruah et al. show that no deterministic algorithm can achieve a constant competitive ratio for the scheduling problem of maximizing the number of jobs completed before their deadlines. Kalyanasundaram and Pruhs [15] show that for every instance at least one of two deterministic algorithms is constant competitive, and thus they provide a randomized algorithm which is constant competitive. However, it is not clear whether and how their results carry over to the more general class of dial-a-ride problems.

The WHAM has also been investigated by Irani, Lu and Regan [10] under the name "dynamic traveling repair problem". The authors give two deterministic algorithms for the $\text{WHAM}_{T,N}$ in general metric spaces with competitive ratios that are formulated in terms of the diameter of the metric space. Their ratios translated into the notation used in this paper and restricted to the uniform metric space are $\frac{3T}{T-2}$ and $4\left\lceil\frac{2T}{T-1}\right\rceil\left(\left\lceil\frac{2T}{T-1}\right\rceil+1\right)$, respectively. We improve these results in several ways.

	upper bound	lower bound
$L \leq T/4$	1	1
$L = T$	$3T + 1$	$\max\{T + 1, \lfloor 3T/2 \rfloor\}$
$L > T$	—	"∞" (same as on $\mathbb{R}_+$ and $\mathbb{R}$)

Table 1
Competitiveness results for the WHAM$_{T,N=1}$ on the truncated line $[-L, L]$.

	upper bound	lower bound
$T \geq 2$	$\frac{\lceil \lfloor T/2 \rfloor + T \rceil}{\lfloor T/2 \rfloor} \in [3, 5]$ (see Figure 3 for a plot)	2
$1 < T < 2$	$2N$	2
$T = 1$	2 (for all t_j integral) $2N$ (for general t_j)	2 (for all t_j integral) $2N$ (for general t_j)

Table 2
Competitiveness results for the WHAM$_{T,N}$ on the uniform metric space.

1.2 Our contribution.

The contributions of this paper are twofold. First, we provide complexity results for the offline problem OFFLINE-WHAM. We derive a dynamic program for the OFFLINE-WHAM on unweighted graphs with integral release times and deadlines, which runs in time $\mathcal{O}(nm(T+m)(\Delta+1)^{2T})$, where n is the number of nodes in the space, m denotes the total number of moles and Δ is the maximum degree. The algorithm runs in polynomial time, if $(\Delta+1)^{2T}$ is bounded by a polynomial in the input size. We complement our solvability result by NP-hardness results.

Our main contribution lies in the analysis of the *online* problem WHAM. We show that no deterministic algorithm for the WHAM on the line can achieve a constant competitive ratio. This unfortunate situation remains true even if one allows randomization.

If the line is restricted to a finite interval $[-L, L]$, the situation changes at the moment where $L \leq T$ (for $L > T$ the same lower bounds as in the case of the unbounded line apply). Here, we give a deterministic algorithm with competitive ratio $3T + 1$ (see Table 1).

From the viewpoint of the *whack-a-mole* player, the situation on the uniform

metric space is better than on the line. Our results for this case are summarized in Table 2. We conclude our study of online algorithms by showing how our results extend to the case of multiple servers.

Our results improve and extend those given in [10] in the following ways: For the line segment $[-T, T]$ and the uniform metric space with $T = 1$ our algorithms are the first competitive ones, since the bounds of [10] cannot be applied. Moreover, for the uniform metric space we decrease known competitive ratios substantially. For instance, for popup duration $T = 2$, our algorithm IWTM achieves a competitive ratio of 3, while the results in [10] yield a ratio of 80. Surprisingly all of our competitiveness results are obtained by simple ("folklore") algorithms.

In terms of lower bounds, the paper [10] shows that there is a metric space in which no deterministic algorithm can achieve a constant competitive ratio. We show that this results is already true on the real line and against more restricted adversary models.

2 The complexity of offline whack-a-mole

In this section we investigate the complexity of the offline problem OFFLINE-WHAM where all moles and their respective release dates are known in advance. We first give a polynomial-time algorithm for a special class of the problem. Then, we show that OFFLINE-WHAM is NP-hard on the line.

In this section we slightly diverge from the notation used for the online problem in allowing more general deadlines $d_j \geq t_j$ for the requests than just $d_j = t_j + T$, where T is the popup duration. In this more general context T will denote the maximum popup duration of any mole. We also allow a weight $n_j \geq 0$ to be associated with request r_j. The goal of the problem becomes to maximize the weight of the whacked moles.

2.1 When whacking is easy

We consider the following scenario: The metric space $M = (X, d)$ has n points and is induced by an undirected unweighted graph $G = (V, E)$ with $V = X$, i.e., for each pair of points from the metric space M we have that $d(x, y)$ equals the shortest path length in G between vertices x and y. We also assume that for each mole the release date $t_j \geq 1$ and the deadline d_j are integers.

Theorem 1 *Suppose that the metric space is induced by an unweighted graph of maximum degree Δ. Then, the* OFFLINE-WHAM *with integral t_j and d_j can*

be solved in time $\mathcal{O}(nm(T+m)(\Delta+1)^{2T})$*, where* $T := \max_{1\leq j\leq m}(d_j - t_j)$ *is the longest time a mole stays above the ground;* n *denotes the number of vertices in the graph and* m *is the total number of requests.*

PROOF. The time bound claimed is achieved by a simple dynamic programming algorithm. Let $0 < t_1 < t_2 < \cdots < t_k$ with $k \leq m$ be the (distinct) times where moles show up. We set $t_0 := 0$.

The idea for a dynamic programming algorithm is the following: For each relevant point t in time and each vertex $v \in V$ we compute the maximum number of moles caught subject to the constraint that at time t we end up at v. Essentially the only issue in the design of the algorithm is how one keeps track of moles that have been whacked "on the way". The key observation is that for any time t that we consider the only moles that need to be accounted for carefully are those ones that have popped up in the time interval $[t-T, t]$. Any mole that popped up before time $t - T$ will have disappeared at time t anyway. This allows us to use a limited memory of the past.

Given a vertex v, a *history track* is a sequence $s = (v_1, v_2, \ldots, v_k = v)$ of vertices in G such that for $i = 1, \ldots, k$ we have $d(v_i, v_{i+1}) = 1$ whenever $v_i \neq v_{i+1}$. We define the time-span of the history track s to be $\bar{d}(s) = k$. The history track s encodes a route of starting at vertex v_1 at some time t, walking along edges of the graph and ending up at v at time $t + \bar{d}(s)$ with the interpretation if $v_i = v_{i+1}$ we remain at vertex v_i for a unit of time. Notice that in an unweighted graph with maximum degree at most Δ, there are at most $(\Delta+1)^L$ history tracks of length $L \in \mathbb{N}$ ending at a specific vertex v.

Given the concept of a history track, the dynamic programming algorithm is straightforward. For $t \in \{t_0, \ldots, t_k\}$, $v \in V$ and all history tracks s, with $\bar{d}(s) = \min(t, T)$, ending in v at time t, we define $M[t, v, s]$ to be the maximum number of moles hit in any solution that starts in the origin at time 0, ends in v at time t, and follows the history track s for the last $\bar{d}(s)$ units of time.

The values $M[0, v, s]$ are all zero, since no mole raises its head before time 1. Given all the values $M[t, v, s]$ for all $t = t_0, \ldots, t_{j-1}$, we can compute each value $M[t_j, v, s]$ easily.

Assume that $t_j \leq t_{j-1} + T$. Then, from the history track s we can determine a vertex v' such that v' must have been at vertex v' a time t_{j-1}. This task can be achieved in time $\mathcal{O}(T)$ by backtracking s. The value $M[t_j, v, s]$ can now be computed from the $\mathcal{O}((\Delta+1)^T)$ values $M[t_{j-1}, v', s']$ by adding the number of moles whacked and subtracting the number of moles accounted for twice. The latter task is easy to achieve in time $\mathcal{O}(T+m)$ given the history tracks s and s'. Hence, the time needed to compute $M[t_j, v, s]$ is $\mathcal{O}((T+m)(\Delta+1)^T)$.

It remains to treat the case that $t_j > t_{j-1} + T$. Let $t := t_{j-1} + T$. Notice that no mole can be reached after time t and before time t_j, since all moles released no later than t_{j-1} will have disappeared by time t. Any solution that ends up at vertex v at time t_j must have been at some vertex v' at time t. We first compute the "auxiliary values" $M[t, v', s']$ for all $v' \in V$ and all history tracks s by the method outlined in the previous paragraph. Then, the value $M[t_j, v, s]$ can be derived as the maximum over all values $M[t, v', s']$, where the maximum ranges over all vertices v' such that v can be reached by time t_j given that we are at v' at time t and given the histories s and s' (which must coincide in the relevant part).

Since the dynamic programming table has $\mathcal{O}(nm(\Delta+1)^T)$ entries, the total time complexity of the algorithms is in $\mathcal{O}(nm(T+m)(\Delta+1)^{2T})$. $\square$

The above dynamic program can easily be adjusted for metric spaces induced by weighted graphs with integral edge weights. Each edge e is then replaced by a path of $w(e)$ vertices, where $w(e)$ denotes the length of edge e. The time bound for the above procedure becomes then $\mathcal{O}(\bar{n}m(T+m)(\Delta+1)^{2T})$, where $\bar{n} = n + \sum_{e \in E}(w(e) - 1)$. Hence, whenever $(\Delta+1)^{2T}$ is pseudo-polynomially bounded, OFFLINE-WHAM can be solved in pseudo-polynomial time on these weighted graphs.

2.2 When whacking is hard

It follows from Theorem 1 that OFFLINE-WHAM can be solved in polynomial time if $(\Delta+1)^{2T}$ is bounded by a polynomial in the input size. On the other hand, the problem on a graph with unit edge weights, all release times zero and all deadlines equal to n, the number of holes, contains the Hamiltonian Path Problem as a special case. Thus, it is NP-hard to solve, see e.g. [14].

Another special case of the OFFLINE-WHAM is obtained when at most one mole is in a hole at a time, the metric space is the line and release dates as well as deadlines are general. Tsitsiklis [16] showed that on this metric space the traveling salesman or repairmain problem with general time windows constraints is NP-complete. This implies that the OFFLINE-WHAM on the line with general release dates and deadlines is NP-hard.

In his proof, Tsitsiklis uses the fact that the length of the time windows may vary per request. This raises the question whether OFFLINE-WHAM on the line with uniform popup durations is NP-hard. In the following theorem we show that this is the case if one allows arbitrary weights to be associated with the moles.

Theorem 2 OFFLINE-WHAM *on the line is NP-hard even if the time moles stay above ground is equal for all moles, i.e.,* $d_i - t_i = d_j - t_j = T$ *for all requests* r_i, r_j.

PROOF. We show the theorem by a reduction from PARTITION, which is well known to be NP-complete to solve [11,7]. An instance of PARTITION consists of n items $a_i \in \mathbb{Z}^+$, $i = 1, \ldots, n$, with $\sum_i a_i = 2B$. The question is whether there exists a subset $S \subset \{1, \ldots, n\}$, such that $\sum_{i \in S} a_i = B$.

Given an instance of PARTITION, we construct an instance I_{WHAM} for OFFLINE-WHAM, with $m = 3n$ requests. Let $B = \frac{1}{2}\sum_i a_i$ and $K = B + 1$. The time each mole stays above ground is $T = 2B$. There are $2n$ requests r_i^+ and r_i^-, $i = 1, \ldots, n$ where $r_i^\pm$ is released at time $(2i-1)K$ and has deadline $(2i-1)K + T$. The position of r_i^+ is $K + a_i$ with weight $K + a_i$, and the position of r_i^- equals $-K$ with weight K. Finally, there are n requests r_i^0 in the origin, where r_i^0 is released at time $2iK$, has deadline $2iK + T$, and weight K.

We claim that at least $2nK + B$ moles can be whacked if and only if I is a YES-instance for PARTITION.

Let S be a partition of I, i.e., $\sum_{i \in S} a_i = B$. Then whacking the moles of requests in the order $(r_1^{\alpha_1}, r_1^0, \ldots, r_n^{\alpha_n}, r_n^0)$, where $\alpha_i = +$ if $i \in S$ and $\alpha_i = -$ if $i \notin S$, is feasible and yields the desired bound, as tedious computation can show.

Suppose conversely that there exists a route for the whacker such that it reaches at least $2nK + B$ moles. Notice that as the locations of the holes of requests r_i^+ and r_i^- are at least $2K > 2B$ apart, the whacker can whack at most one of these requests. The moles of requests r_i^+ and r_i^- pop up after time $t_{i-1}^+ + T$, and therefore the whacker cannot catch the moles of request r_{i-1}^+ and r_i^+ at the same time. The same is true for requests r_{i-1}^0 and r_i^0. Suppose the whacker moves to the hole of r_i^+ or r_i^- after first whacking the moles of r_i^0. The earliest possible arrival time in the mole is at least $2iK + K = (2i+1)K$ and by this time the moles of r_i^+ and r_i^- have gone down again. Hence, when whacking r_i^0 and either r_i^+ or r_i^-, the request r_i^+ or r_i^- need to be whacked before r_i^0. Not whacking the moles of r_i^0 or none of r_i^+ and r_i^-, results in a tour in which at most $(2n-1)K + 2B < 2nK + B$ can be caught. Therefore, the whacker needs to reach all moles popping up in the origin and for each i it also needs to whack all moles of either r_i^+ or r_i^-. Hence, by the above considerations we know that when at least $2nK + B$ moles are whacked, the whacker needs to hit first the moles of r_i^+ or r_i^- and then those of r_i^0 before going to the hole of request r_{i+1}^+ or r_{i+1}^-.

Let $S = \{i : \text{moles of } r_i^+ \text{ are whacked}\}$ be the set of requests served in the

positive part of the line. We claim that $\sum_{i\in S} a_i = B$. Obviously $\sum_{i\in S} a_i \geq B$ since the number of moles whacked is at least $2nK + B$. Suppose that $\sum_{i\in S} a_i > B$ and let $S' \subseteq S$ be the smallest subset of S such that if $i, j \in S$ with $i < j$ and $j \in S'$ then $i \in S'$ and $\sum_{i\in S'} a_i > B$ and let $k = \max S'$. Then $\sum_{i\in S'\setminus\{k\}} a_i \leq B$. The whacker leaves the origin for request r_k^+ at time $2(k-1)K+2\sum_{i\in S'\setminus\{k\}} a_i \leq 2(k-1)K+2B < t_k^0$. The next time the whacker reaches the origin is $2kK + \sum_{i\in S'} a_i > 2kK + 2B$ and by then the moles of request r_k^0 have gone under ground. Hence, it cannot reach the moles of request r_k^0 and is not able to whack $2nK + B$ moles. □

3 Whack-a-mole on the line

In this and the following section we investigate the existence of competitive algorithms for the WHAM. Our lower bound results are not only established for the standard adversary, the optimal offline algorithm, but also for the more restricted adversaries. We stress that our competitiveness results hold for the stronger standard adversary.

3.1 How well we can't whack

The optimal offline algorithm is often considered as an adversary, that specifies the request sequence in a way that the online algorithm performs badly. Besides the ordinary adversary that has unlimited power, there exist several adversaries in the literature that are restricted in their power.

The *non-abusive adversary* of [13] is defined on the line, and it may only move into a certain direction if there is still a pending request in this direction. For WHAM we extend this definition by the restriction that the adversary may only move in the direction of a request that it can reach before the deadline of this request or it may *go home*, i.e., it may move back to the origin. A natural extension to the uniform metric space studied in Section 4 is to require that the adversary only moves on a direct shortest path to a pending request whose deadline can be met.

The following theorem gives a lower bound on the half line $\mathbb{R}_+$ which clearly implies the same bound for the complete line $\mathbb{R}$ (we remark that for the complete line a simpler proof with a request sequence of only one mole can be given).

Theorem 3 *Let $T \geq 0$ be arbitrary. No deterministic online algorithm can achieve a constant competitive ratio for* $\text{WHAM}_{T,N}$ *on the half line $\mathbb{R}_+$ even*

against a non-abusive adversary. This result continues to hold even if $N = 1$.

PROOF. We prove the theorem for popup duration $T = 1$. This proof can be extended to general T by multiplying each position and release time with T. Moreover, our construction only uses a mole-per-hole-limit of $N = 1$.

For any integral constant $c > 2$, we show that there exists a request sequence on which the adversary whacks at least c times as many moles as any deterministic online algorithm. This implies the theorem. The adversarial sequence consists of at most three parts.

σ_1: In the first part a mole is released at each integral time point t in hole $t + 1$.
σ_2: At each integral time point, a mole pops up in hole 0.
σ_3: At each integral time point t, a mole is released in hole $\hat{t} + t + 1 - \bar{t}$.

The sequence starts at time $t = 0$ with σ_1. Let $\hat{t} \leq c$ be the first integral point in time at which the position of the algorithm's whacker at time t $p_{\text{ALG}}(t) \neq t$, or $\hat{t} = c$ if no such $t \leq c$ exists.

If $p_{\text{ALG}}(\hat{t}) \neq \hat{t}$, then the adversary continues with subsequence σ_1 up to time at least $c\hat{t}$. As $p_{\text{ALG}}(\hat{t}) < \hat{t}$, the online algorithm cannot reach any of the moles released at or after time $\hat{t}$, and catches at most $\hat{t}$ moles. The adversary, on the other hand, can whack, by always moving to the right, all moles and serves at least $c\hat{t}$ requests.

If $p_{\text{ALG}}(\hat{t}) = \hat{t}$, i.e., $\hat{t} = c$, the adversary stops subsequence σ_1, after time $\hat{t} - 1$, and continues the sequence with subsequence σ_2 beginning at time $\hat{t}$. Let $\bar{t} \leq c^2 + c + 1$ be the first time $c < t < c^2 + c + 1$ at which $p_{\text{ALG}}(t) = 1$, or $\bar{t} = c^2 + c + 1$ in no such time $c < t < c^2 + c + 1$ exists.

If $\bar{t} = c^2 + c + 1$, then the sequence stops at this time. The algorithm has not whacked any of the moles released in subsequence σ_2, and thus has served at most $\hat{t} = c$ moles. The adversary, by staying in the origin, has reached all moles of subsequence σ_2, and thus killed at least c^2 moles.

If $\bar{t} < c^2 + c + 1$, the adversary stops the subsequence σ_2 at time $\lceil \bar{t} - 1 \rceil$ and continues with σ_3 starting at time $\bar{t}$. As $p_{\text{ALG}}(\bar{t}) = 1 < \hat{t}$, the algorithm cannot reach any of the requests released in the third subsequence, nor has it served any of the requests in σ_2. Hence, it has killed at most c moles. The adversary, by moving to the right during the first subsequence, and remaining in hole $\hat{t}$ during the second one, can reach all moles of the third subsequence, as well as all moles of the first one. By continuing σ_3 for at least $c^2 - c$ time units, the adversary whacks at least c^2 moles. $\square$

The negative result of Theorem 3 above raises the question whether randomized algorithms can perform better.

Theorem 4 *For any $T \geq 0$, no randomized algorithm achieves a constant competitive ratio for* $\text{WHAM}_{T,N}$ *on the line $\mathbb{R}$ against an oblivious adversary. The result remains true for $N = 1$.*

PROOF. Suppose for the sake of a contradiction that there exists a c-competitive randomized online algorithm for some constant c. Let $K = \lceil c \rceil + 1$, and consider the holes $x_i = i(2T+1)$, for $i = 0, 1, \ldots, K-1$. The adversarial sequence consists of only one mole, released at time $\hat{t}$. Let p_i denote the probability that the randomized whacker is within distance T of hole x_i at time $\hat{t} = (K-1)(2T+1)$, As the distance between the holes is more than $2T$, these probabilities sum up to $\sum_{0 \leq i \leq K-1} p_i \leq 1$. Therefore, there is at least one hole x_i where the algorithm's whacker is in reachable distance with probability $p_i \leq 1/K$. At time $\hat{t}$ the adversary releases one mole in this hole x_i. While the adversary certainly catches that mole, the expected value for the algorithm is at most $1/K < 1/c$ which is a contradiction to the assumption of a c-competitive algorithm. □

The lower bound results above suggest to restrict the metric space further. In the sequel we consider the truncated line, $[-L, L]$. Before we embark on lower and upper bound proofs, let us rule out the easy cases. If $L > T$, then no constant competitive ratio can be achieved as a very simple one-mole-sequence shows: suppose the whacker of an online algorithm is in the origin or on the left of it at time T then release one mole in a position larger than T. On the other hand, if $L \leq T/4$, then a trivial algorithm which continuously moves between the end points of the line segment is able to reach each request in time and is therefore optimal. Hence, the only interesting case is $T/4 < L \leq T$.

We consider the problem on the restricted line $[-T, T]$ with unit distances between holes, that is, on $[-T, T] \cap \mathbb{Z}$. Observe that the dynamic program proposed in Section 2.1 solves the related offline-problem efficiently for constant popup duration T and integral release dates. In the following theorem we assume for ease of notation that T is integral. The result can be easily transferred to non-integral values on the cost of at most 1 by replacing T by $\lfloor T \rfloor$ and adjusting the release dates.

Theorem 5 *Let $T \in \mathbb{N}$. No deterministic online algorithm for the* $\text{WHAM}_{T,N}$ *on the line segment $[-T, T] \cap \mathbb{Z}$ can achieve a competitive ratio less than $\max\{NT + 1, N(\lfloor 3T/2 \rfloor)\}$ even against a non-abusive adversary.*

PROOF. At time 0 in each boundary position, T and $-T$, one mole is released. If an online algorithm does not catch a mole by time T then the

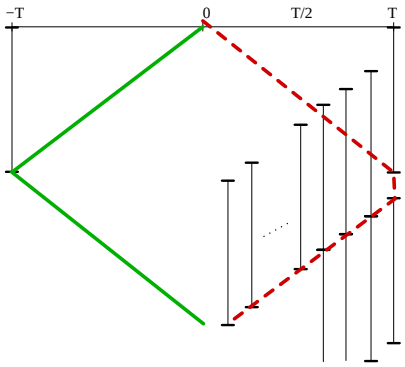

Fig. 1. Lower bound instance for any det. online algorithm OL on the truncated line $[-T, T]$; dashed line: adversary's tour, solid line: OL's tour]. Each request is represented by a vertical line between the release date and deadline.

instance stops. Otherwise, assume w.l.o.g. that it started heading towards $-T$ (the movement must be started at time 0). Then, at time $t = 1$ another $N-1$ moles are released in hole T. Moreover, at each of the times $t = 2, \ldots, T$, N moles get released in holes $T+1-t$ and at time $t = T+1, \ldots, \lfloor 3T/2 \rfloor$ N moles appear in holes $2T+1-t$. If at any time, the algorithm's whacker changes its direction, the sequence stops and the algorithm cannot catch any mole. The construction is illustrated in Figure 1.

A simple calculation shows that no online algorithm is able to catch any of the moles in $[1, T] \cap \mathbb{Z}$ (hence it whacks only a single mole in $-T$) while the adversary whacks all moles on the positive part of the line. Thus, $N(\lfloor 3T/2 \rfloor)$ is a lower bound on the competitive ratio.

The fact that $NT + 1$ is another lower bound on the competitive ratio is even easier to establish. After two initial moles, with weight 1, released at time 0 in $-T$ and T, a second wave of T moles, each with weight N, is released at time T in $1, 2, \ldots, T$ (this assumes that the algorithm moves to the left initially). □

3.2 How well we can whack

As mentioned in the previous section, WHAM on the truncated line $[-L, L]$, with $L \leq T/4$ or $L > T$ are trivially easy or have been shown to be hopeless. In this section, we only consider the line segment $[-T, T]$, for which we have shown a $\max\{NT + 1, N\lfloor 3T/2 \rfloor\}$ lower bound on the competitive ratio. For this case, we analyze the following algorithm which is probably folklore; it can be found also under different names like reopt or optimal [1,8].

Replan (rp)
At any moment in time, compute an optimal route on all pending requests,

ending in the origin. Change the current route if and only if another route allows to whack more moles.

Theorem 6 *Algorithm* RP *achieves a competitive ratio of* $3T+1$ *for* $\text{WHAM}_{T,N=1}$ *on the line segment* $[-T,T] \cap \mathbb{Z}$.

PROOF. Assume that RP is not c-competitive; we prove the theorem by yielding a contradiction for $c \geq 3T+1$. Let $\text{ALG}(\sigma)$ denote the number of served requests for an algorithm ALG on a sequence σ. Denote σ as a smallest sequence for which $\text{RP}(\sigma) < \frac{1}{c}\text{OPT}(\sigma)$. Partition σ into $\sigma = \sigma' \cup \sigma_c$, where σ_c consists of the last c requests. Note, that $\text{RP}(\sigma') \leq \text{RP}(\sigma)$ since at any release date t_j of a request j in σ the *old* route of RP from time $t_j - \epsilon$ is still a feasible route. Then,

$$\text{RP}(\sigma') \leq \text{RP}(\sigma) < \frac{1}{c}\ \text{OPT}(\sigma) \leq \frac{1}{c}\ (\text{OPT}(\sigma') + c) \leq \text{RP}(\sigma') + 1.$$

Due to the integrality of $\text{RP}(\sigma)$ and $\text{RP}(\sigma')$, we know that $\text{RP}(\sigma) = \text{RP}(\sigma')$. By definition of RP, that means that the algorithm does not change its route for any request in σ_c and it does not whack any of those moles. On the other hand, the optimal offline algorithm must serve all requests in σ_c. Otherwise we could remove unwhacked requests from σ_c without changing the routes or solution values of RP and OPT and thus, the sequence σ was not a smallest sequence satisfying $\text{RP}(\sigma) < \frac{1}{c}\ \text{OPT}(\sigma)$

In the remainder we show that RP serves at least one request of σ_c, for $c \geq 3T+1$, which is a contradiction to the previous observations, and thus, we disprove the assumption that RP were not c-competitive for $c \geq 3T+1$.

Let t_{max} be the latest release date of moles in the sequence σ' whacked by RP. We denote the time and position of the last mole whacked by RP by C_ℓ and p_ℓ, respectively. Assume w.l.o.g. that $p_\ell \geq 0$ (the other case is symmetric). Note, that $C_\ell \leq t_{max} + T$, and the time when the whacker returns in the origin is $C_\ell + p_\ell \leq t_{max} + 2T$.

By definition of σ_c, all requests $r_j \in \sigma_c$ have release dates $t_j \geq t_{max}$. If there is any request released at or after RP 's return to the origin, then the whacker is able to catch it, which leads to the contradiction. Therefore, we assume that $t_{max} \leq t_j < C_\ell + p_\ell$ for all $r_j \in \sigma_c$.

After time C_ℓ, there is no reachable pending mole from σ' for RP. Hence, if there is a request $r_j \in \sigma_c$ at a distance of at most T from p_ℓ with $t_j \geq C_\ell$, then RP catches at least one of these moles. This holds at least for all points on the nonnegative halfline. Hence, in holes of the interval $[0,T]$ exluding the hole p_ℓ cannot be more than one mole per hole in sequence σ_c, which sum up

to at most T moles. On the negative part of the line, at most two moles per hole can be released in the time interval $[t_{max}, C_\ell + p_\ell)$ of length less than $2T$.

Hence, the number of moles in σ_c is not more than $3T$, and for any $c \geq 3T+1$ we have a contradiction. This completes the proof of a competitive ratio of $c = 3T + 1$ for RP. $\square$

We note that the above result directly generalizes to an upper bound of $3NT + 1$ on the competitive ratio of RP for a general mole-per-hole limit of N.

4 Whack-a-mole on the uniform metric space

Recall that the uniform metric space is induced by a complete graph with unit edge weights. The number of vertices in this graph is n. Observe that for popup duration $T < 1$ trivially no deterministic algorithm can be competitive against the standard adversary. In case of the non-abusive adversary, the situation is trivial again, since the adversary can never catch any mole except for those in the origin.

4.1 How well we can't whack

We remark that a lower bound of 2 for the WHAM on the uniform metric space has been derived in [10]. Our construction uses fewer nodes in the metric space and, more important, there is no positive amount of time where more than one request is available at a single hole. Also, note that the lower bounds are shown against the most restricted adversary, the non-abusive one.

Theorem 7 *Let $n \geq 3T + 2$, that is, $T \leq (n-2)/3$. No deterministic online algorithm for the* $\text{WHAM}_{T,N=1}$ *can achieve a competitive ratio smaller than* 2 *against a non-abusive adversary.*

PROOF. The idea for our instance is the following: we make sure that at any integral time $t \geq T$ two moles have their deadline and a new mole is released in one of the respective holes, such that an optimal offline algorithm can whack two moles per time unit but an online algorithm catches at most one.

At each time $t = 0, \ldots, T-1$ the adversary releases two moles: one in position $p_1(t) = 2t + 1$ and the other in $p_2(t) = 2t + 2$. At time $t = T, T+1, \ldots$ three

moles are released: two moles are released in empty holes $p_1(t)$ and $p_2(t)$ and the third mole, either, in $p_3(t) = p_2(t-T)$ if ALG is in $p_1(t-T)$ at time t, or, in $p_3(t) = p_1(t-T)$, otherwise. Note that at time t, at most $3T$ moles have deadline at least t, and as $n \geq 3T+2$, there are at least two holes left with no moles at time t. $\square$

In case of $N \geq 1$ the above lower bound can be improved:

Theorem 8 *No deterministic online algorithm for the* $\text{WHAM}_{T=1,N}$ *has a competitive ratio less than* $2N$*, even against a non-abusive adversary.*

PROOF. After an initial step, a non-abusive adversary ADV constructs a sequence consisting of phases such that in each phase it whacks at least $2N$ times as many moles as an online algorithm ALG does. Each phase starts at a time t when the adversary arrives in a hole. We denote by t' the latest deadline of the moles that are in this hole at time t. Note that $t \leq t' < t+1$, since the popup duration is 1. There are two possible positions for ALG to be at time t:

Case (a) ALG is in a vertex point different from the position of ADV;
Case (b) ALG is on an edge.

Moreover, if there are at the beginning of the phase some pending requests released before time t then ALG cannot reach any of them.

In Case (a), two moles are released at time t in holes where neither ALG nor ADV are. If ALG does not immediately go to one of these moles, it cannot whack any of them, whereas the adversary catches one of these moles. Otherwise, at time $\bar{t} = \max\{t', t+1/2\}$ the adversary releases N moles in his current position and N moles in a hole v that is not incident to the edge on which ALG is. Thus, ALG cannot whack any of them. Hence, it whacks at most one mole, whereas ADV reaches $2N$ moles by remaining in his position until time $\bar{t}$ and then moving to v.

In Case (b), ALG is in the interior of an edge and thus, it cannot reach any vertex point which is not incident to this edge by time $t+1$. The adversary releases one mole in a free hole, i.e., a vertex point where no mole is and which is not incident to the edge on which ALG is. Hence, ALG does not whack any mole, and ADV hits one mole.

An initial sequence consisting of two requests in two different holes each releasing a single mole ensures that we end up either in Case (a) or Case (b). This completes the proof. $\square$

Note that in the proof of the above lower bound we use the fact that release dates may be non-integral. As we will see in the next section, this restriction is essential, because for integral release dates we are able to provide a 2-competitive algorithm.

For the sake of completeness we conclude this section of lower bounds with a brief consideration of randomized algorithms. We can easily extend the deterministic lower bound in Theorem 7 to a lower bound for randomized algorithms by blowing up the number of possible holes. Instead of releasing at each time t a mole in each of two free holes, we release moles in $k \geq 2$ free holes. We argue that for at least one of these holes the probability that the online server is in this hole at time $t+T$ is at most $1/k$. The $(k+1)$st mole at time $t+T$ is released in this hole. Then, the expected number of moles caught by an online algorithm is from time $t=T$ onwards $\frac{1}{k} \cdot 2 + \frac{k-1}{k} \cdot 1 = \frac{k+1}{k}$, whereas the optimal offline algorithm can catch 2 moles.

Corollary 9 *Each randomized algorithm has a competitive ratio of at least* 2 *on the uniform metric space.*

4.2 How well we can whack

In this section we analyze simple algorithms for WHAM and give performance guarantees for the online problem on a uniform metric space.

First Come First Kill (fcfk)
At any time t, move to a hole which contains a request with earliest release date, breaking ties in favor of the point where the most moles are above ground. If none of the moles that are above ground can be killed by the algorithm, then the whacker does not move.

Theorem 10 *Let* $T \geq 1$. *Algorithm* FCFK *is* $2N$*-competitive for the* $\text{WHAM}_{T,N}$ *on the uniform metric space.*

PROOF. Partition the input sequence into maximal subsequences, such that each subsequence consists of requests that are released while FCFK is serving continously, i.e., it is constantly moving between holes. We show that an optimal offline algorithm OPT whacks at most $2N$ times as many moles as FCFK does for each subsequence from which the theorem follows.

Consider such a subsequence σ'. We denote by C_j^{ALG} the time where algorithm ALG whacks request r_j. If r_j is not caught, then we set $C_j^{\text{ALG}} = \infty$.

Define the time at which OPT whacks its last mole of σ' as

$$t_{\max} = \max\{\, C_j^{\mathrm{OPT}} : r_j \in \sigma' \text{ and } C_j^{\mathrm{OPT}} < \infty \,\},$$

where we consider an optimum OPT that whacks its last mole as early as possible.

Moreover, we define $t_{\min}$ such that $t_{\max} - t_{\min}$ is integral and $\min_{j\in\sigma'} t_j \leq t_{\min} < \min_{j\in\sigma'} t_j + 1$. In each interval $(t, t+1]$ for $t = t_{\min}, \ldots, t_{\max} - 1$, FCFK hits at least one mole and OPT cannot whack more than $2N$ moles. It remains to show that the moles which are reached by OPT before $t_{\min}$ can be compensated for by FCFK.

If FCFK whacks its last mole of σ' no later than time $t_{\max}$, then OPT catches at most N moles in the interval $(t_{\max} - 1, t_{\max}]$ since no new request can be released at $t_{\max}$ due to the maximality of the subsequence σ'. Moreover, OPT can kill at most N moles in the interval $(\min_{j\in\sigma'} t_j, t_{\min}]$. Therefore, the number of moles reached by OPT during the period before $t_{\min}$ can be accounted for by the moles caught in the last interval by OPT and thus, sum up to at most $2N$.

On the other hand, if FCFK still whacks a mole from σ' after time $t_{\max}$, the number of moles caught by OPT during the first period is at most N times the number of moles hit by FCFK after $t_{\max}$. □

The following lemma shows that the competitive ratio of $2N$ is tight for FCFK, even if one considers the more restricted adversary:

Lemma 11 *Let $T \geq 1$ be an integer.* FCFK *has no competitive ratio less than $2N$ for the* $\mathrm{WHAM}_{T,N}$ *on the uniform metric space against a non-abusive adversary.*

PROOF. At time $t = 0$, the adversary releases T requests in holes $1, \ldots, T$, each of them with weight 1. At time $t = 1/2$, in hole $2T + 1$, a request is released with N moles. At time t, for $t = 1, \ldots, T - 1$, one request in hole $T + t$ is given with one mole and at time $t + 1/2$ a request with N moles is given in $2T + 1 + t$. At time t, for $t = T, T + 1, \ldots$, one mole is popping up in hole $1 + (T + t - 1) \mod 2T$. And at time $t + 1/2$ two requests are given, each with N moles: one in $2T + 1 + (t \mod T)$ and one in $3T + 1 + (t \mod T)$. This sequence is visualized in Figure 2.

Up to time T, FCFK whacks the moles released at time 0. After time T it moves to the hole with the earliest released request that it can reach. As the requests with N moles are released $1/2$ time unit later than the requests with a single mole, FCFK is not able to whack any of the higher weighted requests. Hence, it

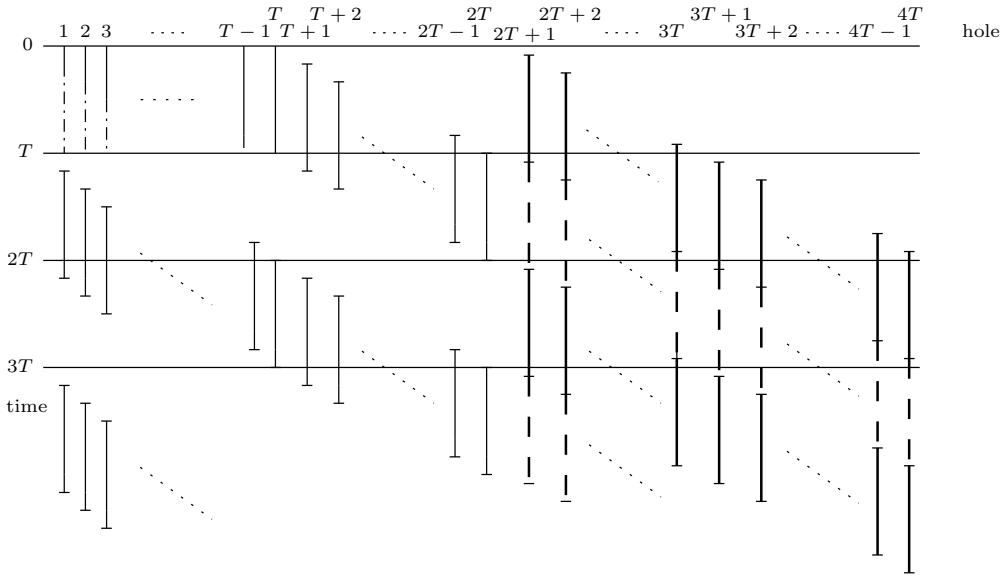

Fig. 2. Lower bound sequence for FCFK. Each request is represented by a vertical line between the release date and deadline. Thick lines illustrate requests with N moles, the thin lines depict requests with single moles. The line segment is dashed after the request has been served by an adversary ADV, and dash-dotted after being served by FCFK and ADV. Notice that from time T onwards, FCFK serves all its requests by their deadlines.

catches in each unit length time interval one mole. In each unit length interval from time $T+1/2$ onwards, there is one hole where a request with N moles has its deadline and a new request with N moles is released. Hence, the adversary ADV can whack $2N$ moles in every unit length time interval after time T. □

Recall that no deterministic online algorithm can be better than 2-competitive (Theorem 7). Hence, by Theorem 10 we know that FCFK achieves a best-possible competitive ratio in the case of a mole-per-hole limit of $N = 1$. For general N but $T = 1$, FCFK is also best-possible by Theorem 8. For the special case of integral release dates and $T = 1$, FCFK obtains a competitive ratio of 2 even for general N.

Theorem 12 *If all release times are integral, then* FCFK *is* 2*-competitive for the* $\text{WHAM}_{T=1,N}$ *on the uniform metric space.*

PROOF. Due to the integral release dates, both, the optimal offline algorithm OPT and FCFK are in holes at integral points in time. Moreover, OPT serves at most two requests released at the same time because of the unit popup duration. FCFK on the other hand, whacks at least the moles of one request released at a certain time and by definition it chooses the request with the highest number of moles. Therefore, it reaches at least half of the moles whacked by OPT. □

Obviously, FCFK's flaw lies in ignoring all requests with a later deadline even though they could contribute with a higher weight to the objective value. In order to overcome this drawback we consider an other algorithm which we call *Ignore and Whack the Most* (IWTM). In this algorithm, we divide the time horizon into intervals of length $l = \lfloor \frac{T}{2} \rfloor$, and we denote these intervals by $I_i = ((i-1)l, il)$, for $i = 0, 1, 2, \ldots, L$, where I_L is the last interval in which moles are whacked. We say that at time t, the *current interval* is the interval I_i for which $t \in I_i$. Note that these intervals only have a positive length for $T \geq 2$.

When formulating the algorithm IWTM we allow the algorithm to whack only a subset of the moles available at a certain hole. Although our problem definition would force all moles at v to be whacked, this condition can be enforced within the algorithm by keeping a "virtual scenario".

Ignore and Whack the Most (iwtm)
At any time when the whacker is in a hole, it moves to the hole with the highest number of pending moles released in the previous interval. Only those moles will be whacked at the hole.

Theorem 13 *Let $T \geq 2$ and $c = \frac{\lceil \lfloor T/2 \rfloor + T \rceil}{\lfloor T/2 \rfloor}$. IWTM is c-competitive for the $\text{WHAM}_{T,N}$ on the uniform metric space.*

PROOF. Let k_i denote the number of moles released in interval I_i, whacked by OPT, and let h_i denote the number of moles whacked by IWTM during interval I_i. Then

$$\text{OPT}(\sigma) = \sum_i k_i, \qquad \text{and} \qquad \text{IWTM}(\sigma) = \sum_i h_i. \tag{1}$$

Moreover, since no moles are released in the last interval I_L, it follows that $k_L = 0$.

First note that IWTM is at integral time points always in a hole. Therefore, during interval I_{i+1} it can visit l holes. If it visits less than l holes, then the number of requests released in interval I_i is less than l. Hence, OPT cannot kill more than h_{i+1} moles of those released in I_i.

Conversely, suppose that IWTM visits exactly l holes during interval I_{i+1}. The optimum can visit at most $\lceil l + T \rceil$ holes of requests released in interval I_i. By definition IWTM serves the l holes with the highest weight of pending requests released in I_i. Therefore, $h_{i+1} \geq (l/\lceil l + T \rceil)k_i$. Hence, by Equations (1), we know that

$$\text{IWTM}(\sigma) \geq (l/\lceil l + T \rceil)\text{OPT}(\sigma).$$

Recall that $l = \lfloor T/2 \rfloor$. $\square$

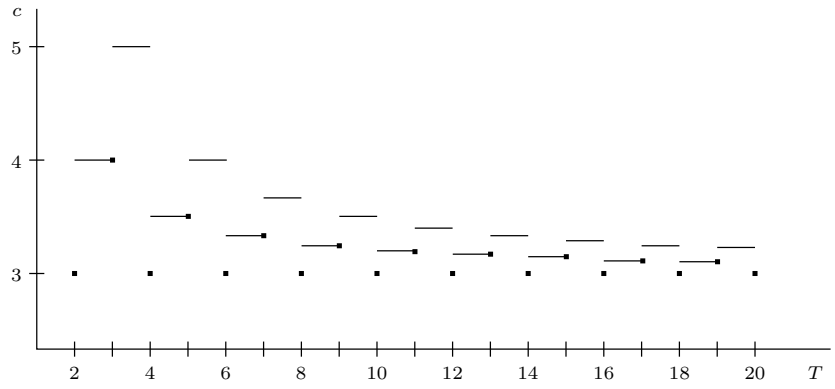

Fig. 3. Competitive ratio for IWTM for $T \in [2, 20]$.

For T ranging from 2 to 20, the values of the competitive ratio of IWTM are depicted in Figure 3.

5 Extensions to Multiple Servers

In this section we briefly discuss the extension of our results to the case of $k \geq 1$ servers.

For the case of the line it is easy to adopt the lower bound results and prove that no deterministic algorithm can achieve a constant competitive ratio for the k-server WHAM on the line.

We proceed to the uniform metric space. Consider the algorithm k-IWTM is the k-server extension of IWTM presented in Section 4.2. For a given sequence σ for the k-server $\text{WHAM}_{T,N}$, we let $\sigma_1, \ldots, \sigma_k$ be disjoint subsequences of σ, such that σ_i contains all requests served by the ith server in the *optimal* solution. We claim that k-IWTM on the sequence σ reaches at least as many moles as the sum of the (single server) IWTM on sequence σ_i.

Lemma 14 $k\text{-IWTM}(\sigma) \geq \sum_{1 \leq i \leq k} \text{IWTM}(\sigma_i)$.

PROOF. Consider all requests released during an interval I_h in all subsequences $\sigma_1, \ldots, \sigma_k$. All these requests are considered by k-IWTM during interval I_{h+1}, and can be served by k-IWTM. As k-IWTM chooses to move to those requests with heighest weight, it will whack at least as many moles released in I_h as the sum of the 1-$\text{IWTM}(\sigma_i)$. □

Theorem 15 *k-IWTM is c-competitive for the k-server $\text{WHAM}_{T,N}$ on the uniform metric space with $c = (\lfloor T/2 \rfloor + T)/\lfloor T/2 \rfloor$.*

PROOF. As the optimal algorithm reaches exactly the sum of all requests reached by the optimal single servers on $\sigma_1, \ldots, \sigma_k$, and 1-IWTM is c-competitive, the theorem immediately follows from the above lemma. □

References

[1] N. Ascheuer, S. O. Krumke, and J. Rambau. Online dial-a-ride problems: Minimizing the completion time. In *Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 639–650. Springer, 2000.

[2] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the on-line traveling salesman. *Algorithmica*, 29(4):560–581, 2001.

[3] S. Baruah, J. Haritsa, and N. Sharma. On-line scheduling to maximize task completions. *The Journal of Combinatorial Mathematics and Combinatorial Computing*, 39:65–78, 2001.

[4] M. Blom, S. O. Krumke, W. E. de Paepe, and L. Stougie. The online-TSP against fair adversaries. *Informs Journal on Computing*, 13(2):138–148, 2001. A preliminary version appeared in the Proceedings of the 4th Italian Conference on Algorithms and Complexity, 2000, vol. 1767 of Lecture Notes in Computer Science.

[5] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[6] E. Feuerstein and L. Stougie. On-line single server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability (A guide to the theory of NP-completeness).* W.H. Freeman and Company, New York, 1979.

[8] M. Grötschel, S. O. Krumke, and J. Rambau, editors. *Online Optimization of Large Scale Systems.* Springer, Berlin Heidelberg New York, 2001.

[9] D. Hauptmeier, S. O. Krumke, and J. Rambau. The online dial-a-ride problem under reasonable load. In *Proceedings of the 4th Italian Conference on Algorithms and Complexity*, volume 1767 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 2000.

[10] S. Irani, X. Lu, and A. Regan. On-line algorithms for the dynamic traveling repair problem. In *Journal of Scheduling*, 7(3):243–258, 2004.

[11] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[12] S. O. Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie. News from the online traveling repairman. *Theoretical Computer Science*, 295(1–3):279–294, 2003. A preliminary version appeared in the Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science, 2001, vol. 2136 of Lecture Notes in Computer Science.

[13] S. O. Krumke, L. Laura, M. Lipmann, A. Marchetti-Spaccamela, W. E. de Paepe, D. Poensgen, and L. Stougie. Non-abusiveness helps: An $O(1)$-competitive algorithm for minimizing the maximum flow time in the online traveling salesman problem. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, volume 2462 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 2002.

[14] D. Poensgen. *Facets of Online Optimization*. Dissertation, Technische Universität Berlin, Berlin, 2003.

[15] K. R. Pruhs and B. Kalyanasundaram. Maximizing job completions online. In *Proceedings of the 6th Annual European Symposium on Algorithms*, volume 1461 of *Lecture Notes in Computer Science*, pages 235–246. Springer, 1998.

[16] J. N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22:263–282, 1992.