How to Whack Moles*

Sven O. Krumke[1**], Nicole Megow[2], and Tjark Vredeveld[1]

[1] Konrad-Zuse-Zentrum für Informationstechnik Berlin, Department Optimization, Takustr. 7, D-14195 Berlin-Dahlem, Germany.
{krumke,vredeveld}@zib.de

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

Abstract. 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.

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.

The *whack-a-mole problem* with popup duration $T \geq 0$ (WHAM$_T$) 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. A server (hammer) moves on the metric space M at unit speed. It

* Supported by the DFG Research Center "Mathematics for key technologies" (FZT 86) in Berlin.

** Research supported by the German Science Foundation (DFG, grant Gr 883/10).

starts in the origin at time 0. Each request $r_j = (t_j, p_j, n_j)$ specifies a *release time* t_j and a point (hole) $p_j \in X$ where n_j moles pop up. 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 WHAM_T.

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. 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 line and the uniform metric space. The 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.

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 (weighted) number of requests served before their deadlines.

The OLTSP has been studied for the objectives of minimizing the makespan [1, 2, 7], the weighted sum of completion times [7, 13], and the maximum/average flow time [9, 14]. 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 completed jobs. Kalyanasundaram and Pruhs [11] 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 in [10] under the name "dynamic traveling repair problem". The authors give two deterministic algorithms for the WHAM_T 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 for the case of the uniform metric space. 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. Moreover, for $T = 1$ our algorithms are the first competitive ones, since the bounds of [10] cannot be applied. The paper [10] also shows that there is a metric space in which no algorithm can achieve a constant competitive ratio. We show that this results is already true on the real line.

Popup duration	upper bound	lower bound
$T \geq 2$	$\frac{\lceil \lfloor T/2 \rfloor + T \rceil}{\lfloor T/2 \rfloor} \in [3, 5]$ (see Figure 2)	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 1. Competitiveness results for the WHAM$_T$ on the uniform metric space where each mole stays T units of time above ground. Here, N denotes the maximum number of moles (possibly stemming from different requests) that are peeking out of the same hole, simultaneously, for a positive amount of time.

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 number 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 for some special cases of the OFFLINE-WHAM.

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. From the viewpoint of the *whack-a-mole* player, the situation is much better on the uniform metric space. Our results for this case are summarized in Table 1. In particular, we provide the first competitive algorithm for short popup durations $T = 1$ and substantially decrease the known competitive ratio for $T \geq 1$. Our competitiveness results are complemented by lower bounds on the competitive ratio of arbitrary deterministic algorithms. The upper bounds hold against the most powerful adversary, whereas the lower bounds are shown against the *non-abusive adversary* [14], which is the most restricted adversary that we consider.

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 and on the star graph. 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.

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 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 $t = t_0, \dots, t_{j-1}$, 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 then 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 [15].

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. Then this problem is also NP-hard by a reduction from PARTITION, as mentioned in [6].

The first case we consider is the weighted version of OFFLINE-WHAM on the line where multiple moles remain above ground for a fixed time.

Theorem 2. OFFLINE-WHAM *on the line is NP-hard even if all moles stay above ground is equal for all moles, i.e., $d_i - t_i = d_j - t_j$ 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 [12, 8]. An instance of PARTITION consists of n items $a_i \in \mathbb{Z}^+$, $i = 1, \dots, n$, with $\sum_i a_i = 2B$. The question is whether there exists a subset $S \subset \{1, \dots, n\}$, 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 $2m$ requests r_i^+ and r_i^-, $i = 1, \dots, m$ 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 m 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 $(r_1^{\alpha_1}, r_1^0, \dots, r_n^{\alpha_n}, r_n^0)$, 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. $\square$

Our next hardness result concerns the case of a star graph.

Theorem 3. OFFLINE-WHAM *is NP-hard on a star graph with arbitrary edge lengths. This result holds, even if at any moment in time at most one mole is peeking out of hole.*

Proof. Let $(a_1, \ldots, a_n)$ be an instance for PARTITION and let $B = \frac{1}{2}\sum_i a_i$. We construct a star graph with $n+1$ leaves. The length of the center to leaf i is equal to a_i, for $i = 1, \ldots, n$ and the length of the $(n+1)$st leg is equal to 1. The request sequence is as follows. At time $t = 0$, a single mole is released in each of the leaves $j = 1, \ldots, n$. The popup duration of each mole is equal to $4B + 2$. At time $2B + 1$ a mole appears in leaf $n + 1$ with zero popup duration. Finally, at time $4B + 2$ a mole pops up in the center, also with zero popup duration.

If we have a YES-instance of PARTITION, i.e., there exists a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = B$, the whacker can catch all moles on the following route. First she whacks the moles in the leaves $i \in S$ in an arbitrary order and arrives at the center at time $2B$. Then, she whacks the mole in leaf $n+1$ after which all the remaining leaves can be visited and the whacker returns in the center at time $2 + \sum_i a_i$, at which she can whack the mole that pops up there.

If there exists a route in which the whacker can reach all moles, then by time $2B + 1$ she needs to be in leaf $n + 1$. Therefore, no later than time $2B$ she needs to be in the center. Let S be the set of moles whacked before this time. After whacking the mole in leaf $n + 1$ and returning to the origin, there is still $2\sum_i a_i - 2B$ time left to whack all the unwhacked moles and return in the center. Therefore, it must hold that $\sum_{i \in S} a_i = B$. $\square$

3 Whack-a-mole on the line

In this and the following section we investigate the existence of competitive algorithms for the WHAM. Our results are not only established for the standard adversary, the optimal offline algorithm, but also for the more restricted adversaries of [4, 14], which we recall briefly.

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 *fair adversary*, introduced by [4], may move at any time only within the convex hull of all requests released so far. An even more restricted adversary is the *non-abusive adversary* of [14]. This adversary 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 restriction that the adversary may only move in the direction of a request that it can reach before the deadline of this request. A natural extension to other metrics is an adversary that may only move on a direct path to a pending request, which deadline can be met.

Theorem 4. *Let $T \geq 0$ be arbitrary. No deterministic online algorithm can achieve a constant competitive ratio for* WHAM_T *on the line.*

Proof. Consider an online algorithm ALG and assume w.l.o.g. that its position at time T is $p_{\text{ALG}}(T) \leq 0$. The sequence consists of one mole, which pops up at time T at position $T + 1$. As the adversary can be at position T by time T, it has killed the mole by time $T + 1$. ALG, on the other hand, can not be in $T + 1$ before time $2T + 1$ and by then the mole has gone under ground again. □

In the proof above the adversary abuses its power in the sense that it moves to a point where a mole will pop up without revealing this information to the online whacker. Theorem 4 can be extended to both, the fair and the non-abusive adversary as is shown in the following theorem.

Theorem 5. *Let $T \geq 0$ be arbitrary. No deterministic online algorithm can achieve a constant competitive ratio for the* WHAM$_T$ *on the line against a non-abusive adversary.*

Proof. Consider an arbitrary deterministic online algorithm. At time 0 two moles appear: one in T and the other in $-T$, both going down at time T. If the online whacker does not reach any mole by time T then the sequence stops. Otherwise, we assume w.l.o.g. that the online whacker is in position $p_{\text{ALG}}(T) = -T$ at time T. ADV is at that time in position T. Then, from time $t = T$ onwards a request is given in $t + 1$ at each integral time $T, T + 1, T + 2, \ldots$. The adversary can whack all these moles, whereas the online whacker is not able to meet any of the deadlines. □

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 $T < 1$ trivially no deterministic algorithm can be competitive against the standard or fair 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 of those defined in the previous section.

Theorem 6. *Let $n \geq 3T + 2$, that is, $T \leq (n - 2)/3$. No deterministic online algorithm for the* WHAM$_T$ *can achieve a competitive ratio smaller than 2 against a non-abusive adversary.*

Proof. At each time $t = 0, \ldots, T - 1$ the adversary releases two moles: one in $p_{t,1} = 2t + 1$ and the other in $p_{t,2} = 2t + 2$. At time $t = T, T + 1, \ldots$ three moles are released: two moles are released in empty holes $p_{t,1}$ and $p_{t,2}$ and the third mole, either, in $p_{t,3} = p_{t-T,2}$ if ALG is in $p_{t-T,1}$ at time t, or, in $p_{t,3} = p_{t-T,1}$, 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.

ALG cannot whack more than one mole per time unit, whereas ADV can kill two moles at a time from time $t = T$ onwards. □

In the sequel the maximum number of moles available at a single hole will play a crucial role. We define N to be the maximum number of moles peeking out of the same hole for a positive amount of time. Notice that with this definition, there might be a moment in time, where in fact $2N$ moles are above ground at the same place. Consequently, an algorithm might whack up to $2N$ moles in a single step.

Theorem 7. *If at most N moles can be in the same hole for a positive amount of time, then any deterministic online algorithm for the* WHAM$_1$ *against a non-abusive adversary has a competitive ratio no less than $2N$.*

Proof. After an initial step, a non-abusive adversary constructs a sequence consisting of phases such that in each phase it whacks at least $2N$ times as many moles as 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. □

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

4.2 How well we can whack

In this section we propose 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 8. *Let $T \geq 1$. Consider the* WHAM$_T$ *with at most N moles peeking out of one hole for a positive amount of time.* FCFK *is $2N$-competitive, which is tight.*

Proof. Partition the input sequence into maximal subsequences, such that each subsequence consists of requests that FCFK serves continously, i.e., it is constantly moving between holes. We show that 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

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

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 $t_{\max}$.

Lemma 1 implies that the competitive ratio of $2N$ is tight for FCFK. □

Lemma 1. *Let $T \geq 1$ be an integer. Consider the* WHAM$_T$ *with at most N moles peeking out of one hole for a positive amount of time.* FCFK *has no competitive ratio less than $2N$ 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 1.

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 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, ADV can whack $2N$ moles in every unit length time interval after time T. □

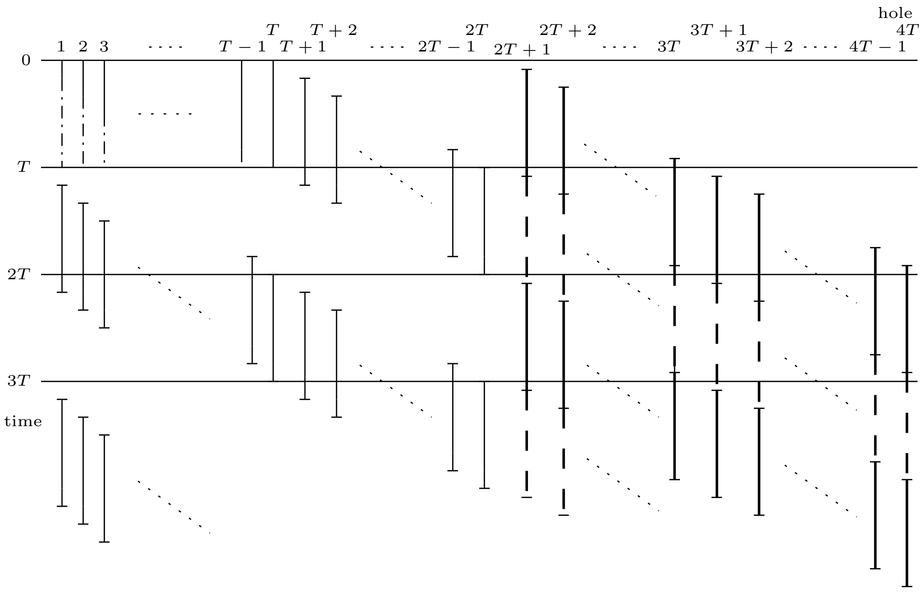

Fig. 1. Lower bound sequence for FCFK. Each request is represented by a vertical line between the release date and deadline. Thick lines illustrate requests with N moles, the thin lines depict requests with single moles. The line segment is dashed after the request has been served by ADV, and dash-dotted after being served by FCFK and ADV. Notice that from time T onwards, FCFK serves all its requests by their deadlines.

Recall that no deterministic online algorithm can be better than 2-competitive (Theorem 6). Hence, by Theorem 8 we know that FCFK is optimal in the unweighted setting, i.e., at most one mole can be peeking out of a single hole for a positive amount of time. Also, in the weighted case with $T = 1$, FCFK is optimal by Theorem 7. For the special case of integral release dates and $T = 1$, FCFK obtains a competitive ratio of 2 even in the weighted setting.

Theorem 9. *Consider the* WHAM$_1$ *where each mole stays* $T = 1$ *time unit above ground.* FCFK *is* 2*-competitive if all release dates are integral.*

Proof. Due to the integral release dates, both, 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 the optimal offline algorithm. □

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 in 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 at a hole, it moves to the hole with the highest number of pending moles released before the current interval that can still be reached in time. Only those moles will be whacked at the hole.

Theorem 10. *Let* $T \geq 2$ *and* $c = \frac{\lceil \lfloor T/2 \rfloor + T \rceil}{\lfloor T/2 \rfloor}$. IWTM *is* c*-competitive for the* WHAM$_T$.

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 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 at or before time il. 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 $l = \lfloor T/2 \rfloor$. For T ranging from 2 to 20, the values of the competitive ratio are depicted in Figure 2. □

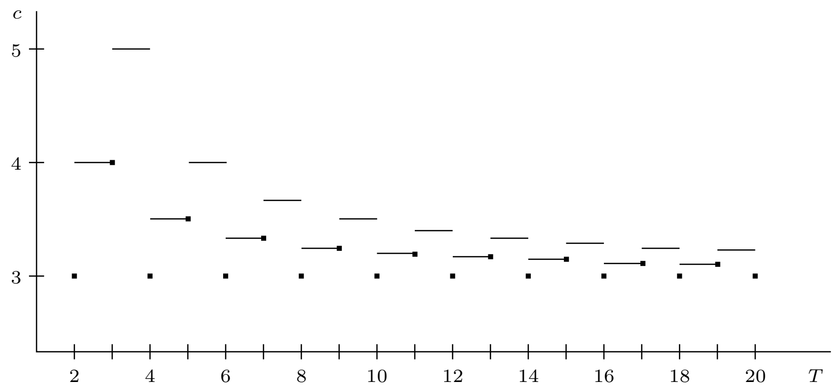

Fig. 2. Competitive ratio for IWTM for $T \in [2, 20]$.

5 Concluding remarks

We have provided the first competitive algorithms for the WHAM on the uniform metric space for small popup duration and improved known competitiveness results for larger values. For the case of non uniform popup durations a natural extension of the FCFK algorithm would be the *Whack the Wimp* algorithm: always move to a hole containing a request with earliest deadline (breaking ties just as in FCFK). If all popup durations are at least 1, then using the same analysis as for FCFK we can show the same bounds on the uniform metric space. In the case that there are popup durations less than 1, no deterministic algorithm can be constant competitive.

Our lower bound results show that the situation on the real line is hopeless in terms of competitiveness, at least for deterministic algorithms. This raises the question whether randomized algorithms can do better.

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.
5. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
6. W. E. de Paepe, J. K. Lenstra, J. Sgall, R. A. Sitters, and L. Stougie. Computer-aided complexity classification of dial-a-ride problems. *Informs Journal on Computing*, 2003. To appear.
7. E. Feuerstein and L. Stougie. On-line single server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001.
8. 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.
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 *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 517–524, 2002.
11. B. Kalyanasundaram and K. R. Pruhs. 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.
12. 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.
13. S. O. Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie. News from the online traveling repairman. *Theoretical Computer Science*, 295:279–294, 2003.
14. 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.
15. D. Poensgen. *Facets of Online Optimization*. Dissertation, Technische Universität Berlin, Berlin, 2003. Submitted.