RAIRO-Theor. Inf. Appl. 43 (2009) 189–207
DOI: 10.1051/ita:2008024

Available online at:
www.rairo-ita.org

ON THE POWER OF RANDOMIZATION FOR JOB SHOP SCHEDULING WITH k-UNITS LENGTH TASKS *

Tobias Mömke[1]

Abstract. In the job shop scheduling problem k-*units*-J_m, there are m machines and each machine has an integer processing time of at most k time units. Each job consists of a permutation of m tasks corresponding to all machines and thus all jobs have an identical dilation D. The contribution of this paper are the following results; (i) for $d = o(\sqrt{D})$ jobs and every fixed k, the makespan of an optimal schedule is at most $D + o(D)$, which extends the result of [3] for $k = 1$; (ii) a randomized on-line approximation algorithm for k-*units*-J_m is presented. This is the on-line algorithm with the best known competitive ratio against an oblivious adversary for $d = o(\sqrt{D})$ and $k > 1$; (iii) different processing times yield harder instances than identical processing times. There is no 5/3 competitive deterministic on-line algorithm for k-*units*-J_m, whereas the competitive ratio of the randomized on-line algorithm of (ii) still tends to 1 for $d = o(\sqrt{D})$.

Mathematics Subject Classification. 68W20, 68W25.

1. Introduction

In job shop scheduling, there are m machines and d jobs. Each job consists of an individual sequence of tasks, and each task is associated with exactly one machine. Each machine can only process the task of one job at a time. Before a task of a job J can be performed, all preceding tasks of J have to be completed. Minimizing the makespan of job shop scheduling is known to be NP complete. Willamson *et al.* [8] showed that not even a $(5/4 - \varepsilon)$-approximation algorithm for any ε can exist.

Keywords and phrases. On-line algorithms, randomization, competitive ratio, scheduling.

* *Supported in part by SNF grant 200021-107327/1.*

[1] Department of Informatics, ETH Zurich, ETH Zentrum, 8092 Zürich, Switzerland;
tobias.moemke@inf.ethz.ch

Article published by EDP Sciences

In this paper, we consider a generalized version of the well studied job shop scheduling problem *unit-*J_m with m different machines [6] in order to compare randomization with determinism. We call the generalized problem *k-units-*J_m, because we allow a processing time of up to k time units for each task.

The problem *k-units-*J_m is similar to acyclic job shop scheduling (see [2]) and relates to finding optimal schedules for routing packages with variable length. Its on-line version turned out to be very suitable for demonstrating the power of randomization. In *k-units-*J_m, we consider jobs consisting of m tasks. Each machine processes exactly one task of each job. Therefore, all jobs contain the same tasks and the jobs only differ by the permutation of the tasks. As in the general job shop scheduling problem, each machine processes only one task at a time and each job must be executed on the machines in the given order without preemption. The time to process a task is a positive integer number of up to k time units and determined only by its corresponding machine. The objective is to minimize the makespan over all feasible solutions, *i.e.*, the time to process all jobs on all machines.

Obviously, a lower bound on the makespan is the time to process one job without delays, *i.e.*, the sum of the processing times of all machines, known as the dilation D.

For $m \geq 3$, the problem *unit-*J_m is NP-hard, see [7]. This directly implies the NP-hardness of *k-units-*J_m. In the on-line versions of the problems, the fixed order of tasks that a job has to complete is hidden for the on-line algorithm and revealed one by one. Thus, each time the next task is known, but not the following ones.

Hromkovič *et al.* showed in [3] that for $d = o(\sqrt{m})$ the makespan of an optimal schedule of *unit-*J_m is at most $m + 2d\sqrt{m}$. They also presented a randomized on-line algorithm for *unit-*J_m with $d = o(\sqrt{m})$ which is $(1 + 2d/\sqrt{m})$-competitive against an oblivious adversary and has linear runtime. We generalize the results of [3] for the problem *k-units-*J_m. Furthermore we analyze upper and lower bounds on the makespan for some special cases which enables us to compare randomization with determinism. The contribution of this paper can be summarized as follows:

(i) we provide a randomized $(1 + d \cdot (k + 3)/(2\sqrt{D}))$-competitive on-line approximation algorithm k-OLR$_m$ for *k-units-*J_m that runs in linear time. For $d = o(\sqrt{D})$ jobs, its competitive ratio amounts to $1 + o(1)$. This demonstrates the power of randomness for *k-units-*J_m because even for *unit-*J_m we do not know any off-line approximation algorithm with an approximation ratio tending to 1;

(ii) for the case $d = 2$ we give an on-line algorithm that solves *k-units-*J_m for any instance consuming at most $(3D + k)/2$ unit steps. For every deterministic on-line algorithm, we present a hard instance with two jobs that inherently exploits the structural properties of *k-units-*J_m. The length of the computed schedules is at least

$$D + \frac{D \cdot (k-1) + 1}{2k - 1}.$$

If k is large and $D \gg k$, the difference between the upper and lower bound is negligible;

(iii) we show that for every deterministic on-line algorithm for k-*units*-J_m there is an input instance with three jobs such that the resulting makespan is at least
$$D + \frac{2D}{3} - D/(3k-2);$$

(iv) we show that there is no 5/3 competitive deterministic on-line algorithm for k-*units*-J_m whereas for a constant k and $o(\sqrt{D})$ jobs, the competitive ratio of the randomized on-line algorithm k-OLR$_m$ tends to 1. This demonstrates that for k-*units*-J_m randomization is very powerful.

2. Preliminaries

An input instance of k-*units*-J_m consists of m machines $\sigma_1, \ldots, \sigma_m$, their processing times $p_1, p_2, \ldots, p_m$, where $p_j \in \{1, 2, \ldots, k\}$ for $j = 1, 2, \ldots m$, and d jobs $J_1, \ldots, J_d$ for some $d \geq 2$ represented as permutations of $(1, 2, \ldots, m)$.

For all integers i, j, we call the jth task of the ith job $\tau_{i,j}$. Each task defines unambiguously exactly one machine that processes the task. For convenience, we will use the notation $\tau_{i,j}$ also for the number of the machine which processes $\tau_{i,j}$. This way, $\sigma_{\tau_{i,j}}$ is the machine that processes the jth task in Job i. For the processing time of tasks, we will use the shortened notation $p_{i,j} := p_{\tau_{i,j}}$ for all i, j.

The tasks have to be processed without preemption. Thus, while some task $\tau_{i,j}$ is processed, the corresponding machine $\sigma_{\tau_{i,j}}$ is occupied by job J_i for $p_{i,j}$ consecutive time units and its processing may not be interleaved by another job.

The time to finish a job J_i is determined by its dilation $D = \sum_{j=1}^{m} p_j$ and the waiting time caused by occupied machines.

Figure 1 shows an example of an input instance I of k-*units*-J_m with two jobs where $k = 3$ and $m = 6$ holds.

We can assume that there are no schedules where all jobs are waiting simultaneously. Therefore, it is obvious that for every input instance the makespan of any schedule is at least D and, by sequentially processing all jobs, at most $D \cdot d$.

We use a geometrical representation of input instances that is based on [1] and [3]. First, we restrict our description to input instances with only two jobs. Afterwards we will formally define the representation of arbitrary instances.

Given an input instance of k-*units*-$J_m$with two jobs and dilation D, we employ a $D \times D$ grid for the geometrical representation.

The axes of the grid represent the jobs. Each column represents one time-unit of the first job J_1 and each column belongs to exactly one task. Thus, for every $j = 1, 2, \ldots, m$, task $\tau_{1,j}$ corresponds to $p_{1,j}$ consecutive columns. Analogously, job J_2 is represented in the rows of the grid, see Figure 2a.

Let $\tau_{1,j} = \tau_{2,j'}$, *i.e.*, $\tau_{1,j}$ and $\tau_{2,j'}$ have to be processed on the same machine. Then the columns of $\tau_{1,j}$ and the rows of $\tau_{2,j'}$ intersect in the grid. The intersection defines a compound of grid squares, *i.e.*, 1×1-squares in the grid. We call such a compound an obstacle. The term obstacle is motivated by the fact that each

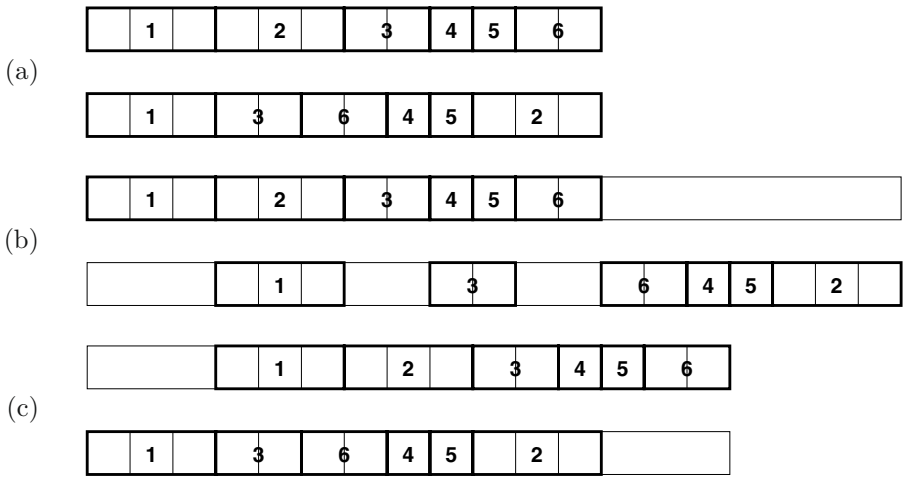

FIGURE 1. (a) An input instance for k-*units*-J_m with two jobs, 6 machines. (b) A valid schedule for (a). (c) The minimum schedule for (a).

point within an obstacle represents a conflict, because the corresponding machine cannot process both jobs simultaneously. Note that each column and each row contains exactly one obstacle.

We define a directed graph of the grid. The vertex set of the graph consists of all vertices of the grid. The arcs are all orthogonal edges of the grid from the left to the right and bottom-up that are not within an obstacle. Additionally, for each grid square which is not within an obstacle, we add an arc from the lower left to the upper right. A valid schedule is represented by a path in the graph starting from the lower left vertex and ending in the upper right one. The goal is to find a shortest path between those vertices. The instance of Figure 1 represented geometrically is shown in Figure 2. Figure 2a shows the rows and columns defined by the two jobs. The graph of the grid and the paths corresponding to Figures 1b and 1c are illustrated in Figures 2b and 2c respectively.

Note that in the geometrical representation, the processing of a machine may pause. Obviously, the length of a schedule, where a job J waits while being processed by some machine σ is equivalent to the length of a schedule where J waits after being processed by σ and the delays of all other jobs are the same as before (especially no job starts processing on σ). Therefore, every path in Graph$(G_D^d(I))$ can be efficiently transformed into a valid schedule for I.

Now we define formally the geometrical representation for an arbitrary number of jobs. A d-dimensional instance I of k-*units*-J_m with dilation D is represented as a d-dimensional $D \times D \times \ldots \times D$ grid $G_D^d(I)$ that is a subgrid of an infinite d-dimensional grid. Each axis of the grid is associated with one job of the instance. For $i = 1, 2, \ldots, d$, the ith axis of $G_D^d(I)$ is labelled by $(\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,m})$. As in

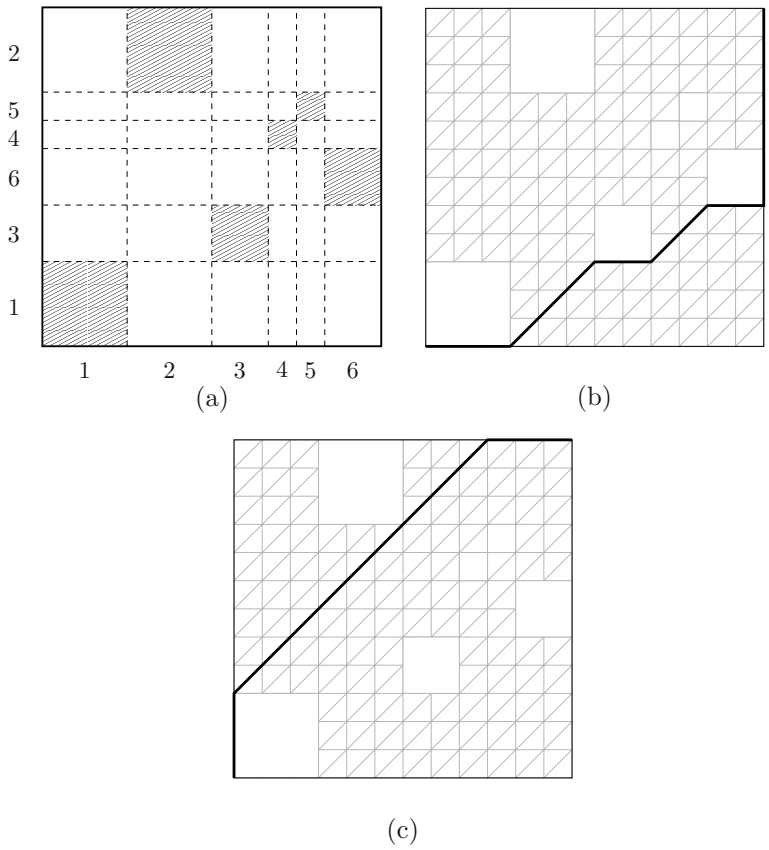

FIGURE 2. The geometrical representation of the schedules from Figure 1.

the two dimensional case, the processing time of each machine is represented by its length on the axis and there are no gaps between the machines. Thus, the axes define the advance of the jobs.

Consider a machine σ_j, $j \in \{1, 2, \ldots, m\}$. The intersection of σ_j on two different axes defines a $p_j \times p_j$ square in the grid, independent of the other axes and this way $D^{d-2} \cdot p_j^2$ grid hypercubes are specified. All hypercubes specified by the $\binom{d}{2}$ combinations of axes form one **obstacle** for σ_j (see Fig. 3).

For every input instance I we assign the grid $G_D^d(I)$ to the directed graph $\text{Graph}(G_D^d(I)) = (V, E)$, where V consists of all vertices of $G_D^d(I)$ (described by their coordinates) and E consists of all arcs $(u, v) \in V \times V$, where $u := (u_1, u_2, \ldots, u_d)$ and $v = (v_1, v_2, \ldots, v_d)$ such that $u \neq v$, v is not inside of an obstacle, the arc (u, v) does not traverse an obstacle, and for $l = 1, 2, \ldots, d$ either $v_l = u_l$ or $v_l = u_l + 1$. This way, every valid schedule for an input instance I of *k-units-*J_m can be described as a path through $\text{Graph}(G_D^d(I))$ (see Fig. 4).

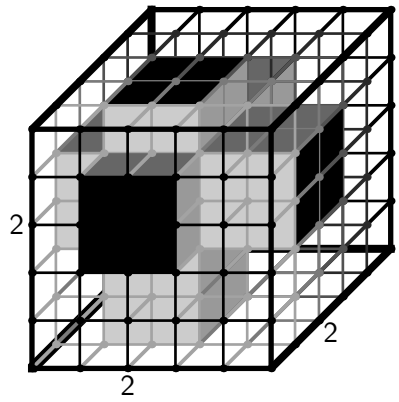

FIGURE 3. An example for a 2-obstacle in G_5^3.

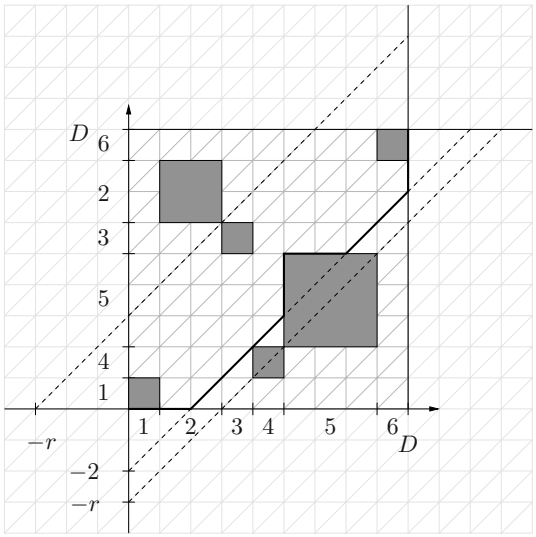

FIGURE 4. An example for the geometrical representation of a two dimensional instance G_9^2 of *k-units-*J_m. The arcs of Graph(G_9^2) are represented by gray lines. The thick line represents the schedule $S(\Delta(0,-2))$. The dotted lines represent some diagonals from $\mathcal{D}(r)$.

For an axis ρ we define the **ρ-border** as $\{(v_1, v_2, \ldots, v_d) \in \{1, 2, \ldots, D\}^d \mid v_\rho = D\}$. Further, let t be the number of synchronized time-steps. With each time-step, a schedule S advances one arc in the graph of the grid. Then $S(t) \in \{1, 2, \ldots, D\}^d$ denotes the position in the grid reached by S after t time-steps and $|S|$ denotes the length of S, *i.e.*, the number of arcs traversed in the graph. Considering the

set of arcs in S up to position $S(t)$, then $\mathbf{diag}(\boldsymbol{t})$ denotes the number of diagonal – and $\mathbf{ort}(\boldsymbol{t})$ the number of orthogonal steps at time t; a step is diagonal, if all jobs advance simultaneously and it is orthogonal if exactly one job advances. For an axis ρ, $\mathbf{ort}_{\boldsymbol{\rho}}(\boldsymbol{t})$ denotes the number of orthogonal steps on ρ at time t. We denote the schedule computed by an algorithm A on input I by $A(I)$. The **main-diagonal** is the diagonal between the extreme corners of the grid.

In on-line problems, one only obtains part of the input and has to process it immediately. After processing, one obtains a new part of the input. Now the fundamental question is, how good an on-line algorithm can be compared to an algorithm that knows the whole input. For this purpose, the **competitive ratio** is used. Let A be an on-line algorithm for a minimization problem and I an input for A. Further let $\mathrm{cost}_A(I)$ be the cost of the computed solution and $\mathrm{Opt}(I)$ the cost of an optimal solution. Then the competitive ratio of A is defined as

$$\mathrm{comp}_A(I) := \mathrm{cost}_A(I)/\mathrm{Opt}_A(I).$$

A is δ-competitive, if $\mathrm{comp}_A(I) \leq \delta$ for every valid input I. For randomized on-line algorithms, we need the expected competitive ratio **Exp-Comp**. Let Z_I be a random variable that measures the cost of the solution computed by a randomized on-line algorithm A. Then

$$\text{Exp-Comp}_A(I) = \frac{\mathrm{E}[Z_I]}{\mathrm{Opt}(I)}.$$

Analogously, A is Exp[δ]-competitive, if $\text{Exp-Comp}_A(I) \leq \delta$ for every valid input I.

3. The effect of long tasks in the randomized on-line algorithm $\mathrm{OLR}_{\mathsf{m}}$

In this section, we will also consider schedules that do not start within the grid. For an instance I, we employ an infinite grid containing $G_D^d(I)$. The graph of the infinite grid is defined in the same way as for the finite one. This way, we can define schedules with initial delays by changing the start of the schedule using negative numbers as coordinates in the grid. Obviously, these initial delays cannot shorten the schedule. We consider start coordinates where exactly one job starts immediately and all other jobs have an initial waiting time between one and r time-units for an integer r. Let $(v_1, v_2, \ldots, v_d)$ be a coordinate in the grid such that $v_l = 0$ for exactly one $l \in \{1, 2, \ldots, d\}$ and $0 > v_b \geq -r$, for all $b \in \{1, 2, \ldots, d\} - \{l\}$. Then we define $\Delta(v_1, v_2, \ldots, v_d)$ as a diagonal in the grid starting at $(v_1, v_2, \ldots, v_d)$ and ending at $(v_1 + D + a, v_2 + D + a, \ldots, v_d + D + a)$, where $a := \|(v_1, v_2, \ldots, v_d)\|_\infty \leq r$. Then $\boldsymbol{\mathcal{D}(r)}$ denotes the set of all diagonals $\Delta(v_1, v_2, \ldots, v_d)$ defined as above for a given r.

Each diagonal $\Delta \in \mathcal{D}(r)$ determines a schedule S_Δ as follows: Let $(v_1, v_2, \ldots, v_d)$ be the starting coordinate of Δ. Then for each l, job J_l is postponed $|v_l|$ time units with respect to the starting time. When the end of the diagonal is reached, all jobs are finished because $i_l + a + D \geq D$ for all l. We modify OLR$_m$ of [3] to deal with k-obstacles.

On-Line Algorithm 1 (k-OLR$_m$)**.**

Input: *The number of jobs d and the length of the axis D are known initially and $d = o(\sqrt{D})$. The machines of the jobs are presented one by one for each job in the order of their occurrences, and in arbitrary order across the jobs.*

Step 1: *Choose uniformly at random a diagonal Δ from $\mathcal{D}(r)$, i.e., generate the start coordinates of Δ at random.*

Step 2: *Apply the schedule S_Δ by avoiding obstacles as they appear. Immediately after avoiding an obstacle, the schedule returns to Δ. Let σ be the machine that forms the obstacle that has to be avoided. Then the obstacle is avoided as follows:*

Step 2.1: *if there is a job that already occupies σ, let this job finish the processing on σ;*

Step 2.2: *machine σ processes the waiting jobs successively.*

The area of a two dimensional k-obstacle corresponds to k^2 1-obstacles. We denote these obstacles as **virtual 1-obstacles**.

Lemma 1. *Let I be an input instance for k-units-J_m with $d = 2$ jobs.*

(i) *In k-OLR$_m$, the number of steps taken to avoid a k-obstacle crossed by the diagonal Δ equals the number of steps used to avoid all of its virtual 1-obstacles that are crossed by Δ.*

(ii) *The number of virtual 1-obstacles in I is at most*

$$\lfloor D/k \rfloor \cdot k^2 + (D \mod k)^2.$$

(iii) *The average delay of k-OLR$_m(I)$ is at most*

$$\frac{k+1}{2} \cdot \sqrt{D} + \frac{1}{2}.$$

Proof.

(i) Let (i, j) be the lower left corner of an k'-obstacle in $G_D^2(I)$ for some $k' \leq k$. W.l.o.g. assume that the schedule reaches some coordinate (i, j'), where $j \leq j' \leq j + k'$. Avoiding the obstacle obviously consumes $2 \cdot (j + k' - j')$ steps. Crossing the obstacle would consume $j + k' - j'$ diagonal steps. If we assume a virtual 1-obstacle for each diagonal step, avoiding all 1-obstacles one after the other consumes also $2 \cdot (j + k' - j')$ steps (see Fig. 5).

(ii) We will show by contradiction that no instance can imply more virtual obstacles than I. Let I' be an input instance of k-units-J_m with $d = 2$ and dilation D with the maximum number of virtual 1-obstacles larger than

$\lfloor D/k \rfloor \cdot k^2 + (D \mod k)^2$. We can assume w.l.o.g., that the machines are ordered by their processing times such that $p_1 \leq p_2 \leq \ldots \leq p_m$ holds. In I', the number of virtual 1-obstacles is $\sum_{l=1}^{m} p_l^2$. If $p_2 = k$, *i.e.*, there is at most one j such that $p_j < k$ in I', then we are obviously done, because there are $\lfloor D/k \rfloor$ machines with processing time k, at most one machine of size $D \mod k$.

Otherwise, $p_1 = \xi < k$ and $p_2 = \xi' < k$. But then there is an instance I'' with $p_3, p_4, \ldots, p_m$ as in I', $p_1 = \xi - 1$, and $p_2 = \xi' + 1$. But this is a contradiction because $\xi^2 + \xi'^2 < (\xi - 1)^2 + (\xi' + 1)^2$.

(iii) We determine the average delay by generalizing the proof of [3]. For simplicity we assume $D = l^2$ for some l. Let $\Delta_i \in \{\Delta(0, i), \Delta(i, 0)\}$.

Because of (i) we only have to consider 1-obstacles. At the border, the schedule S_{Δ_i} uses i additional steps in order to finish both jobs. Observe that the makespan of this schedule is exactly

$$D + i + \text{ the number of 1-obstacles crossed by } \Delta_i$$

because the length of Δ_i is $D - i$ and the schedule uses $2 \cdot i$ steps to leave and to return to Δ_i. Therefore, the delay of the schedule S_{Δ_i} is $i +$ the number of obstacles at Δ_i. Now we have to determine the number of delays. Let $s = D \mod k$ and $t = (D - s)/k$, *i.e.*, there are t k-obstacles and one s-obstacle. Then the number of virtual 1-obstacles is

$$\begin{aligned} t \cdot k^2 + s^2 &\leq t \cdot k^2 + s \cdot k \\ &= (k \cdot t + s) \cdot k \\ &= D \cdot k. \end{aligned}$$

There are

$$\sum_{i=-\sqrt{D}}^{\sqrt{D}} |i|$$

delays to be considered for reaching the diagonals. Then we get

$$\begin{aligned} \frac{\text{number of delays}}{\text{diag}(|S|)} &= \frac{D \cdot k + \sqrt{D} \cdot \left(\sqrt{D} + 1\right)}{2 \cdot \sqrt{D} + 1} \\ &\leq \frac{k+1}{2} \cdot \sqrt{D} + \frac{1}{2} \end{aligned}$$

as an upper bound on the average delay. □

Now we are ready to turn towards instances with any number of jobs. We count the number of possible delays. A label l on an axis in the grid $G_D^d(I)$ of an instance with d jobs determines p_l $d-1$-dimensional subgrids, each consisting of D^{d-1} d-dimensional unit grid cubes. Two different axes labelled by l determine p_l^2 intersections of D^{d-2} d-dimensional grid cubes each.

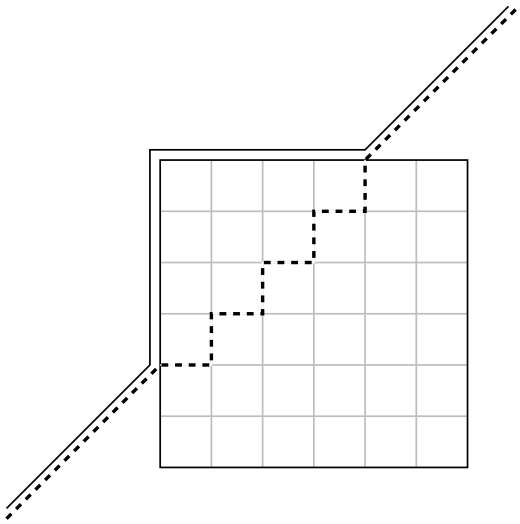

FIGURE 5. The virtual path through the obstacle (the dotted line) has exactly the same length as the path that avoids the obstacle (the plain line).

Lemma 2. *Let σ_l be the lth machine in an instance of k-units-J_m and $r \leq D$. The intersection of every pair of p_l $(d-1)$-dimensional subgrids determined by σ_l causes at most*

$$\frac{p_l + p_l^2}{2} \cdot (d-1) \cdot r^{d-2}$$

delays over all of the $d \cdot r^{d-1}$ diagonals of $\mathcal{D}(r)$.

Proof. We generalize the proof of [3]. Every set of $(d-1)$-dimensional subgrids determined by machine σ_l intersects each of the diagonals of $\mathcal{D}(r)$ in exactly p_l grid cubes. We bound the number of diagonals intersecting the set of $(d-2)$-dimensional subgrids and count the number of 1-obstacles[1] crossed by these diagonals. Similar to [3] we use the relative delay between the bth job and the ath job.

Let the ith position of the ath axis and the jth position on the bth axis be the lower left corner C_j of the p_j-obstacle caused by machine σ_j.

All diagonals that reach C_σ have to cross p_l 1-obstacles. The parallel diagonals of distance ξ, where $\xi < p_l$, have to cross $p_l - \xi$ 1-obstacles. The sum of all 1-obstacles crossed is

$$p_l + \sum_{\xi=1}^{p_l-1} \xi = \frac{p_l + p_l^2}{2}.$$

We count the number of diagonals from $\mathcal{D}(r)$ with the relative delay $j-i$ between the bth and the ath job. Let $\mathcal{D}_n(r)$ contain all diagonals with the nth value

[1]Using Lemma 1 it is sufficient to consider the 1-obstacles that are crossed by the diagonal.

of the starting coordinate equal to 0. Since $\mathcal{D}(r)$ is the union of all $\mathcal{D}_n(r)$ and $\mathcal{D}_\phi(r) \cap \mathcal{D}_\psi(r) = \emptyset$ for $\phi \neq \psi$, where $\phi, \psi \in \{1, 2, \ldots, d\}$ holds, we count the number of such diagonals in $\mathcal{D}_n(r)$ for each n separately.

Let $n' \in \{1, 2, \ldots, d\} - \{a, b\}$. The intersection of $\mathcal{D}_{n'}(r)$ and C_{σ_l} meets all the diagonals $\Delta(c_1, c_2, \ldots, c_d)$, where $c_{n'} = 0$ and $c_b = c_a + j - i$. There are r possible choices for every position from $d-3$ positions of $\{1, 2, \ldots, d\} - \{n, a, b\}$, and at most $r - (j - i) \leq r$ choices for the ath axis. The bth axis is unambiguously determined by the position of a. Thus, we have at most r^{d-2} obstacles in the intersection of C_σ and $\mathcal{D}_b(r)$ with $t_b = 0$ and $t_a = i - j$, where $\Delta(t_1, t_2, \ldots, t_d) \in \mathcal{D}_b(r)$. The number of such diagonals with two fixed coordinates is exactly r^{d-2}. C_σ does not intersect any diagonal from $\mathcal{D}_a(r)$ because the diagonals $\Delta(s_1, s_2, \ldots, s_d)$ in $\mathcal{D}_a(r)$ have $s_a \geq s_u$ for every $u \in \{1, 2, \ldots . d\}$, *i.e.*, the ath job (including the bth job). Thus, C_σ intersects altogether at most

$$(d - 1) \cdot r^{d-2}$$

diagonals.

For each of these diagonals we have to consider at most $(p_l + p_l^2)/2$ delays. Thus,

$$\frac{p_l + p_l^2}{2} \cdot (d - 1) \cdot r^{d-2}$$

bounds the number of delays from above. □

We employ Lemma 2 in order to bound the length of optimal schedules.

Theorem 1. *For every positive integer D and every instance I of the problem k-units-J_m the length of any optimal schedule can be bounded from above by*

$$\mathit{makespan}(I) \leq D + \frac{k + 3}{2} \cdot d \cdot \sqrt{D}.$$

Proof. Each schedule without delays is at most as long as the corresponding diagonal $\Delta(i_1, i_2, \ldots, i_d)$.

We modify the proof of [3]. Since in every diagonal $\Delta(i_1, i_2, \ldots, i_d) \in \mathcal{D}(r)$ exactly one coordinate is zero, the number of diagonals in $\mathcal{D}(r)$ is exactly

$$d \cdot r^{d-1}. \tag{1}$$

Note that one could consider also diagonals whose starting coordinates contain several 0 elements, but this makes the calculation more complex and the achievable gain is negligible.

As in the 2-dimensional case, we calculate the upper bound on the total delay of all $d \cdot r^{d-1}$ schedules. This bound can be obtained as the sum of an upper bound on the number of the lengths of all diagonals and of an upper bound on the number of all delays occurring on these diagonals.

Since the starting coordinates of all diagonals in $\mathcal{D}(r)$ lie on the boundary of the grid and at the end at most r extra diagonal steps are added, the length of each described diagonal is bounded from above by $D + r$.

Because of (1), the sum of the lengths of all diagonals is at most

$$d \cdot r^{d-1} \cdot (D + r). \tag{2}$$

Now we count the number of possible delays. If p jobs have to be processed on one machine σ at the same time, we have to consider a delay for $p-1$ of the jobs. Note that the amount of time that one job already has consumed on σ reduces the delay that we have to consider. Remember that a label σ on an axis determines a set of $(d-1)$-dimensional subgrids of $G_D^d(I)$. We calculate the total number of delays as the sum of the number of delays caused by pairs of $(d-1)$-dimensional subgrids with the same label.

A label σ on an axis determines a $(d-1)$-dimensional subgrid of $G_D^d(I)$. Because of Lemma 1 we get an upper bound on the number of virtual 1-obstacles by assuming every obstacle to be a k-obstacle. Then there are at most D/k machines and the number of schedule delays on all $d \cdot r^{d-1}$ diagonals is at most

$$D/k \cdot \frac{k+k^2}{2} \cdot \binom{d}{2} \cdot r^{d-2} = D \cdot \frac{k+1}{2} \cdot \binom{d}{2} \cdot r^{d-2}.$$

Therefore, the average number of delays per diagonal is at most

$$\left(D \cdot \frac{k+1}{2} \cdot \binom{d}{2} \cdot r^{d-2} \right) / (d \cdot r^{d-1}).$$

We obtain

$$D + 2r + (Dd^2(k+1))/(4r)$$

as a bound on the average makespan of all diagonal schedules. Choosing $r = d\sqrt{D}/2$ we get

$$D + d\sqrt{D} \cdot (k+3)/2. \qquad \square$$

With Theorem 1, the main result of this section is straightforward to obtain.

Corollary 1. *k-OLR$_m$ runs in linear time and its expected competitive ratio is at most*

$$1 + \frac{\frac{k+3}{2} \cdot d}{\sqrt{D}}.$$

For $d = o(\sqrt{D})$ and growing D, Exp-Comp$_{k\text{-}OLR_m}$ tends to one.

Proof. Let I be an input for k-OLR$_m$. Then for every schedule, makespan$(I) \geq D$ and following Theorem 1,

$$\mathrm{E}[|k\text{-OLR}_m(I)|] \leq D + (k+3) \cdot d\sqrt{D}/2.$$

Therefore,

$$\frac{D + (k+3) \cdot d \cdot \sqrt{D}/2}{D} = 1 + \frac{(k+3) \cdot d/2}{\sqrt{D}}$$

is an upper bound on the competitive ratio. $\square$

4. Randomization is powerful for on-line k-*units*-J_m

In this section we compare the expected competitive ratio of k-OLR$_m$ with the best competitive ratio achievable with deterministic on-line algorithms. It is common to treat on-line problems as games played by the algorithm designer against an adversary that knows the on-line algorithm A and, if A is randomized, its probability distribution. Given an on-line algorithm from the algorithm designer, the adversary creates an input instance. Since on-line algorithms compute partial solutions in order to get more input, the input is revealed piece by piece. Thus, in k-*units*-J_m, the adversary chooses the set of tasks and their order within the jobs.

Let σ_j be the jth machine. Then we call a task processed on σ_j a **p_j-task**. We call an obstacle determined by p_j-tasks a **p_j-obstacle**. For simplicity we assume that k is not larger than D. Directly under an obstacle only orthogonal horizontal steps are possible. Based on that, we design the following adversary. The horizontal axis is denoted by ρ_x and the vertical by ρ_y.

Adversary 1 (on-line k-*units*-J_m, $d = 2$)**.**

Input : *An on-line algorithm A for k-units-J_m and the lengths of the tasks.*
Step 1: *If $k \geq D/2$ then place a task of length k on $(0, 0)$ and jump to 7. else place a task τ of length k on ρ_x and place a 1-task on ρ_y. $T := \tau$.*
Step 2: *As long as A takes orthogonal vertical steps and the next task is not specified, place new 1-task on ρ_y. If the distance to the border is exactly the length of T, place task T to the next position instead and jump to 7.*
Step 3: *As soon as A takes a orthogonal horizontal or a diagonal step, place T on the next free position of the y axis.*
Step 4: *Let k' be the length of T. Until A has advanced $k' - 1$ steps on ρ_x or has reached the border fill up every unspecified position with 1-tasks.*
Step 5: *If both borders have a distance of at least k, place a k-task on the next position of ρ_x, $T :=$ this machine and jump to Step 2.*
Step 6: *Place a task of maximum length on the last possible position of ρ_x and set $T :=$ this machine. Do steps 2 and 3.*
Step 7: *Place 1-tasks at all remaining positions.*

In Figure 6, an example of a constructed instance with a corresponding schedule illustrates how the adversary acts.

Lemma 3. *Let A be a deterministic on-line algorithm for k-units-J_m and let I be an input for A constructed by Adversary 1. Furthermore, let S be the schedule computed by A with $S(t) = (i, j)$ being the schedule at time t. If $i = n \cdot (2k - 1)$ holds for $n \in \mathbb{N}$, then*

$$ort_x(t) \geq n \cdot (k - 1).$$

Proof. We distinguish the cases $\mathrm{ort}_y(|S|) = 0$ and $\mathrm{ort}_y(|S|) > 0$.

(i) Let $\mathrm{ort}_y(|S|) = 0$. Then every step advances on ρ_x. We show by induction that $j \leq n \cdot k$ holds. We have to consider all $n \in \mathbb{N} - \{0\}$ such that $n \cdot (k - 1) \leq D$.

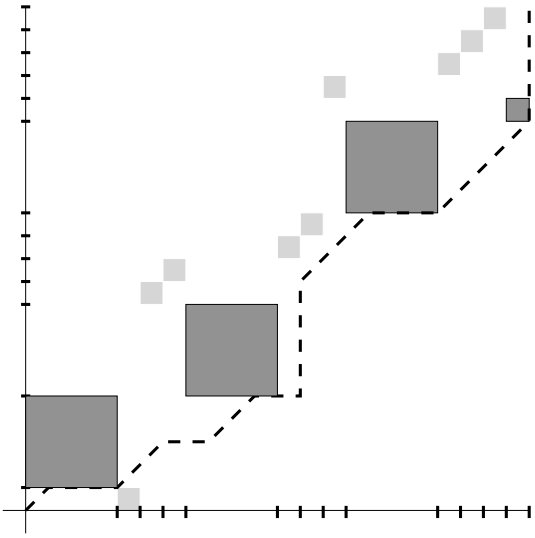

FIGURE 6. A possible schedule constructed by Adversary 1.

Because of the k-obstacle at $(0,1)$, we have $S(k) = (k,0)$ or $S(k) = (k,1)$. The $k-1$ steps after the obstacle can be taken diagonally. Thus,

$$S(2k-1) = (i', 2k-1)$$

with $i' \leq 1 + (k-1) = k$. The next task on ρ_x is of size k. Now let $S((n-1)\cdot(2k-1)) = (i,j)$ and $j \leq (n-1)\cdot k$. Then the adversary requires k-tasks for both jobs such that there is a k-obstacle on $(i,(n-1)\cdot k+1)$. While the vertical way is obstructed by this obstacle, we have

$$S(t) \in \{(t,j') \mid j' \leq (n-1)\cdot k+1\}$$

for $t \leq (n-1)\cdot(2k-1)+k$ and

$$S(t') \in \{(t',j') \mid j' \leq n\cdot k\}$$

for $t' \leq t+k-1 = n\cdot(2k-1)$.

This implies directly

$$\mathrm{ort}_x \geq n\cdot(k-1).$$

(ii) Let $\mathrm{ort}_y(|S|) > 0$. Obviously, orthogonal vertical steps reaching positions below valid schedules of (i) cannot reduce the number of orthogonal horizontal steps. Steps on ρ_x between two k-tasks are not counted in (i). Thus, the only positions to be considered are those that are directly before k-machines on ρ_x above all previous obstacles on ρ_y.

For $n = 0$, $\mathrm{ort}_x(t) \geq n \cdot (k-1)$ is trivially fulfilled. Now let $S(t) = (i, j)$ such that $i = n \cdot (2k-1)$ and j is above all previous obstacles, and let $S(t') = (i', j')$ such that $i' = (n+1) \cdot (2k-1)$. There must be an l such that

$$S(t+l) \in \{(i+l, j+1), i+l-1, j+1)\}.$$

If $i + l \leq D - k$ or $i + l \geq D$, the following $k-1$ steps are orthogonal horizontal. Otherwise, $i'-1 = 2k-1$ and $j'-j \leq k$ which implies that at least $k-1$ orthogonal horizontal steps are required to reach (i', j'). Thus,

$$\mathrm{ort}_x(t') \geq n + 1 \cdot (k-1). \qquad \square$$

The number of orthogonal horizontal steps from Lemma 3 is crucial for the following theorem.

Theorem 2. *For every deterministic on-line algorithm A for k-units-J_m, Adversary 1 constructs an input instance I_A with two jobs such that*

$$|A(I_A)| \geq D + \frac{D \cdot (k-1) + 1}{2k-1}.$$

Proof. Let $n := \lfloor D/(2k-1) \rfloor$ and $l := D \mod (2k-1)$. We analyze the maximum number of diagonal steps. Following Lemma 3, $\mathrm{ort}_x(D)$ is at least $n \cdot (k-1)$. At time $t = n \cdot (2k-1)$, the ratio $\mathrm{ort}_x(t)/(\mathrm{ort}_x(t) + \mathrm{diag}(t))$ is at least

$$\frac{n \cdot (k-1)}{n \cdot (k-1) + (n \cdot (2k-1) - n \cdot (k-1))} = \frac{k-1}{2k-1}.$$

Let $S := A(I_A)$. All steps on ρ_x must sum up to D. Let s be the time such that $S(s) = (D, j)$ for some j. Then $\mathrm{diag}(s) + \mathrm{ort}_x(s) = D$. Thus,

$$\mathrm{ort}_x(|S|) \geq \mathrm{ort}_x(s) \geq D \cdot \frac{k-1}{2k-1} - 1$$

and

$$|S| \geq 2 \cdot D \cdot \frac{k-1}{2k-1} - 1 + \left(D - \left(D \cdot \frac{k-1}{2k-1} - 1 \right) \right) = D + \frac{D(k-1)+1}{2k-1}. \qquad \square$$

On the other hand, we use the following simple deterministic on-line algorithm in order to bound the makespan from above:

On-Line Algorithm 2 (Greedy)**.**

Input: *An input instance I of k-units-J_m with $d = 2$ jobs.*
Rule 1: *Whenever possible take diagonal steps.*
Rule 2: *Prefer orthogonal horizontal step to orthogonal vertical steps.*

Lemma 4. *Let Greedy run on an input instance I with $d = 2$.*

(i) *Greedy runs in linear time.*
(ii) *At least $(D - k)/2$ diagonal steps are taken.*

Before we can prove Lemma 4, we first need the following lemma.

Lemma 5. *Let (i, j) be the lower left corner of a k'-obstacle, $1 \leq k' \leq k$. Then from $v = (i, j + k')$ and $w = (i + k', j)$ either k' diagonal steps are possible or the border is reachable with fewer than k' steps.*

Proof. W.l.o.g. we only consider v. The proof for w is analogous. If $D - \max\{i + k', j\} \leq k'$, then the border is reachable in k' orthogonal steps because the tasks of the jobs cannot overlap and thus no obstacle can be in the way.

Let $D - \max\{i + k', j\} > k'$ and assume that from v, only l diagonal steps can be taken, where $0 \leq l < k'$. Then there must be an obstacle at $(i + k' + l, j + l)$. But this would imply two overlapping tasks for one job which is a contradiction (see Fig. 7). □

Proof of Lemma 4.

(i) The size of the input is $D \cdot d$. In order to determine a diagonal, $d \cdot \lceil \log_2 (d\sqrt{D}/2) \rceil$ random bits are sufficient. Therefore, the first step runs in linear time. The time to execute Step 2 is not longer than the schedule itself which cannot be longer than $d \cdot n$ steps.

(ii) Let S be the schedule calculated by Greedy and let $f : \mathbb{N} \to \mathbb{Q}$ be the function that maps time $t \geq 1$ to $\mathrm{diag}(t)/(\mathrm{ort}(t)+\mathrm{diag}(t))$ and $f(0) := 1/2$.

Let, for an integer n, $\nu(n)$ be the minimum value of t such that $|\{t' \leq t \mid f(t') \geq 1/2\}| = n$ holds. We show by induction on n that, if the distance to the border is $b \geq 2k$, then there is an s with $\nu(n) < s \leq \nu(n)+2k$ such that $f(s) \geq 1/2$. If there is no obstacle at $(0, 0)$, Greedy starts with a diagonal step. Then $f(1) = 1 \geq 1/2$. Otherwise, a k'-obstacle, where $k' \leq k$, is located at $(0, 0)$. Thus, Greedy takes k' horizontal steps. Following Lemma 5, the succeeding k' steps are diagonal. Therefore, $(2k', k')$ is reached within $2k'$ steps and $f(2k') = k'/(k' + k') = 1/2$.

Let $(i, j) = S(\nu(n))$. If a diagonal step is possible, then

$$\begin{aligned} f(\nu(n+1)) &= (\mathrm{diag}(\nu(n)) + 1)/(\mathrm{ort}(\nu(n)) + \mathrm{diag}(\nu(n)) + 1) \\ &\geq f(\nu(n)) \geq 1/2. \end{aligned}$$

Otherwise, a k''-obstacle, where $k'' \leq k$, is located at $(i - c, j)$ or $(i, j - c)$ for $0 \leq c < k''$. Then from (i, j), the following $k'' - c$ steps must be orthogonal and afterwards k'' diagonal steps are possible. Altogether, we get

$$\begin{aligned} f(\nu(n+1)) &= (\mathrm{diag}(\nu(n)) + k'')/(\mathrm{diag}(\nu(n)) + k'' + \mathrm{ort}(\nu(n)) + k'' - c) \\ &\geq (\mathrm{diag}(\nu(n)) + k'')/(2k'' + \mathrm{diag}(\nu(n)) + \mathrm{ort}(\nu(n))) \\ &\geq (\mathrm{diag}(\nu(n)) + k'')/(2 \cdot (\mathrm{diag}(nu(n)) + k'')) \\ &\geq 1/2. \end{aligned}$$

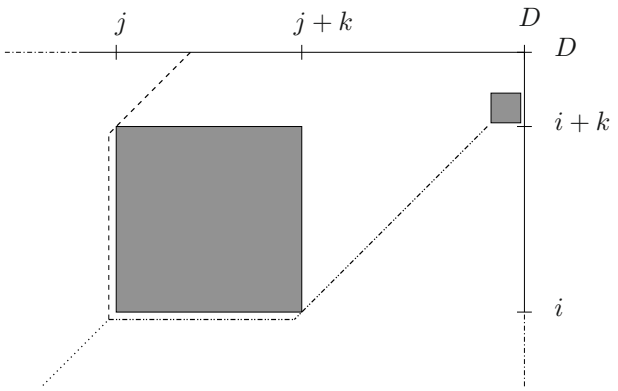

FIGURE 7. As shown in Lemma 5, after a k-obstacle either k diagonal steps are possible (the lower schedule) or a border is reached somewhere (the upper schedule).

Thus, there is an $r \leq 2k$ such that after the first $D - r$ steps $f(r) \geq 1/2$ holds. Thus, at least $(D - 2k)/2$ of the first $D - r$ steps are diagonal. If $r \leq k$, we are done.

Otherwise, a simple induction on r (left to the reader) shows that the number of diagonal steps in the remaining schedule suffices. □

With the number of diagonal steps from Lemma 4 we can easily prove the following theorem.

Theorem 3. *For any input I with $d = 2$, $|Greedy(I)|$ is bounded from above by*

$$\frac{3D}{2} + \frac{k}{2}.$$

Proof. Let $S :=$Greedy(I). From every coordinate (i, j), (D, D) is reachable within $(D - i) + (D - j)$ orthogonal steps. We know from Lemma 4 that $\mathrm{diag}(|S|) \geq (D - k)/2$, and thus

$$\mathrm{ort}(|S|) \leq 2 \cdot (D - (D - k)/2)\,.$$

Altogether,

$$\begin{aligned} |S| &\leq \frac{D-k}{2} + 2 \cdot \left(D - \frac{D-k}{2} \right) \\ &= 2D - \frac{D-k}{2} \\ &= \frac{3D+k}{2}. \end{aligned}$$

□

Lemma 6. *Let A be a deterministic algorithm for on-line k-units-J_m. There is an input instance I with $d = 3$ such that A results in a makespan of at least*

$$D + \frac{2D}{3}\left(1 - \frac{1}{2k-1}\right).$$

Proof. For a given algorithm the adversary constructs the following instance. Let ρ_x, ρ_y, and ρ_z be the three axes in I.

Rule 1: Place all k-tasks as Adversary 1; ρ_z is treated independently like ρ_y.
Rule 2: Place the 1-tasks on ρ_y and on ρ_z such that every second step one of the jobs has to wait (see [4]).

The 1-tasks are either placed on both, ρ_y and ρ_z or in increasing order. This ensures that each time the tasks are available for both axes.

In order to bound the length of the schedule from below, we consider ρ_y and the ρ_z separately; then I_y and I_z are formed by the ρ_x and either ρ_y or ρ_z of I respectively.

For simplicity we assume that $(2k-1)$ divides D. Let $S := A(I)$ with S_y and S_z denoting the schedules corresponding to I_y and I_z respectively. Further let $n := D/(2k-1)$, where n is an integer and t the time when S reaches the ρ_x-border. Following Lemma 3, for S_y and for S_z, $\mathrm{ort}_x(t) \geq n \cdot (k-1)$ and thus $\mathrm{diag}(t) \leq n \cdot k$.

For simplicity, let ort_y refer to S_y and ort_z to S_z.

Orthogonal steps on ρ_y or ρ_z cause delays on ρ_x. At most every second step of $\mathrm{ort}_y(t)$ and $\mathrm{ort}_z(t)$ can be done simultaneously on both axes. Thus, we have to consider at least

$$\frac{4}{3} \cdot \frac{\mathrm{ort}_y(|S|) + \mathrm{ort}_z(|S|)}{2}$$

delays on ρ_x. Thus, we get

$$\begin{aligned} |S| &\geq D + \frac{4 \cdot n \cdot (k-1)}{3} \\ &= D + \frac{2D}{3} \cdot \left(1 - \frac{1}{2k-1}\right). \end{aligned}$$

□

For $d = 3$ dimensions, we combine results from [4] and Adversary 1. We exploit the fact that Adversary 1 can place the 1-obstacles arbitrarily.

Theorem 4. *There is no deterministic on-line algorithm for k-units-J_m with a competitive ratio better than* $5/3$.

Proof. Let A be a deterministic on-line algorithm for k-*units*-J_m. Lemma 6 implies that for A there is a hard instance I such that the resulting schedule is longer than

$$\left(\frac{5}{3} - \varepsilon\right) \cdot D$$

where $0 < \varepsilon < 2/(6k-3)$. On the other hand, Corollary 1 implies that the minimal makespan for I tends to D with growing D. Thus, A is not $\left(\frac{5}{3}-\varepsilon\right)$-competitive. □

5. Conclusions

Based on the algorithm OLR_m from [3] we presented the randomized algorithm k-OLR_m that tends to be 1-competitive for a large D. For on-line k-*units*-J_m with two and three jobs we presented new upper and lower bounds on the makespan and we proved the nonexistence of deterministic on-line algorithms for k-*units*-J_m that are better than 5/3-competitive. Thus, randomization improves the competitive ratio significantly.

Acknowledgements. The author would like to thank Juraj Hromkovič for helpful discussions and the referees for their comments and suggestions.

References

[1] P. Brucker, An efficient algorithm for the job shop problem with two jobs. *Computing* **40** (1988) 353–359.
[2] U. Feige and C. Scheideler, Improved bounds for acyclic job shop scheduling. *Combinatorica* **22** (2002) 361–399.
[3] J. Hromkovič, K. Steinhöfel and P. Widmayer, Job shop scheduling with unit length tasks: bounds and algorithms. ICTCS '01: Proceedings of the 7th Italian Conference on Theoretical Computer Science. *Lect. Notes Comput. Sci.* **2202** (2001) 90–106.
[4] J. Hromkovič, T. Mömke, K. Steinhöfel and P. Widmayer, Job shop scheduling with unit length tasks: bounds and algorithms. *Algorithmic Operations Research* **2** (2007) 1–14.
[5] F.A. Leighton, B.M. Maggs and S.B. Rao, Packet routing and job shop scheduling in O(congestion + dilition) steps. *Combinatorica* **14** (1994) 167–186.
[6] F.A. Leighton, B.M. Maggs and A.W. Richa, Fast algorithms for finding O(congestion + dilation) packet routing schedules. *Combinatorica* **19** (1999) 375–401.
[7] J.K. Lenstra and A.H.G. Rinnooy Kan, Computational complexity of discrete optimization problems. *Annals of Discrete Mathematics* **4** (1979) 121–140.
[8] D.P. Willamson, L.A. Hall, C.A.J. Hurkens, J.K. Lenstra, S.V. Sevast'janov and D.B. Shmoys, Short shop schedules. *Operations Research* **45** (1997) 288–294.

Communicated by J. Hromkovič.
Received August 28, 2007. Accepted April 24, 2008.