A note on semi-online machine covering

Tomáš Ebenlendr[1], John Noga[2], Jiří Sgall[1], and Gerhard Woeginger[3]

[1] Mathematical Institute, AS CR, Žitná 25, CZ-11567 Praha 1, The Czech Republic. Email: ebik,sgall@math.cas.cz.
[2] Department of Computer Science, California State University, Northridge, CA 91330, U.S.A. Email: jnoga@ecs.csun.edu.
[3] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. Email: gwoegi@win.tue.nl.

Abstract. In the machine cover problem we are given m machines and n jobs to be assigned (scheduled) so that the smallest load of a machine is as large as possible. A semi-online algorithm is given in advance the optimal value of the smallest load for the given instance, and then the jobs are scheduled one by one as they arrive, without any knowledge of the following jobs. We present a deterministic algorithm with competitive ratio $11/6 \leq 1.834$ for machine covering with any number of machines and a lower bound showing that no deterministic algorithm can have a competitive ratio below $43/24 \geq 1.791$.

1 Introduction

In the machine cover problem we are given m identical machines and n jobs to be assigned (scheduled) so that the smallest load of a machine is as large as possible.

The motivation for this objective function comes from applications where the jobs correspond to supplies (like fuel tanks) needed to keep the machines alive, and the overall goal is to keep the whole system alive as long as possible. The same objective was studied before for example in [5], where some additional motivation can be found.

Similarly to the classical makespan problem, the ideal schedule is perfectly balanced. Thus the exact solution is NP-hard, and using similar techniques as for makespan scheduling, approximation schemes can be constructed even for uniformly related machines [6, 2, 1, 4].

It is easy to see that in the online setting with jobs arriving one by one, no non-trivial deterministic algorithm is possible [3]. If m jobs with processing times equal to 1 arrive, the algorithm has to assign them to distinct machines, as this may be the whole sequence. Then $m-1$ jobs with processing time m arrive, and the online algorithm achieves objective 1 while the optimum is m.

With this in mind, Azar and Epstein [3] considered semi-online algorithms which are given in advance the value of the optimum. Among other results, they showed that a simple greedy algorithm is $2 - 1/m$ competitive, this is optimal for $m = 2, 3, 4$ for deterministic algorithms, and no semi-online deterministic algorithm for $m \geq 4$ is better than 1.75-competitive.

Our results

We focus on semi-online algorithms for large m. We present a deterministic algorithm with competitive ratio $11/6 \leq 1.834$ for machine covering with any number of machines. This is the first semi-online algorithm whose competitive ratio is strictly smaller than 2.

We also present a lower bound showing that no deterministic algorithm can have a competitive ratio below $43/24 \geq 1.791$. This improves the previous lower bound of 1.75 and is reasonably close to the upper bound.

2 Preliminaries

We are given m machines and n jobs with processing times (or size) $p_j \geq 0$. A *schedule* is an assignment of jobs to machines $S : \{1, \dots, n\} \to \{1, \dots, m\}$.

The *load* of machine i is the sum of the processing times of the jobs assigned to that machine, denoted by $L_i = \sum_{j \in S^{-1}(i)} p_j$. A machine is *$L$-covered* (for a number L) if its load is at least L in the given schedule ($L_i \geq L$).

The objective is to maximize the minimal load of a machine, $\min_i L_i$. An optimal schedule for the given instance I is denoted $OPT(I)$ and its objective value is denoted $L^{\mathrm{OPT}}(I)$.

A semi-online algorithm A is given in advance the value $L^{\mathrm{OPT}} = L^{\mathrm{OPT}}(I)$ (and the value m). Then the jobs of the instance I are scheduled one by one as they arrive, without any knowledge of the following jobs. Its objective on the given instance I is denoted $L^{\mathrm{A}}(I)$. The algorithm is called R-competitive if $L^{\mathrm{OPT}}(I) \leq R \cdot L^{\mathrm{A}}(I)$ for any instance I.

Note that, given the desired competitive ratio R, a semi-online algorithm knows the covering level L^{OPT}/R which it needs to achieve. After some partial sequence, if there exists an assignment with m' of L^{OPT}-covered machines, then the algorithm actually needs to guarantee that it has at least m' of L^{OPT}/R-covered machines. The reason is that the instance can continue with $m - m'$ jobs with $p_j = L^{\mathrm{OPT}}$. Intuitively, this means that if the number of machines is sufficiently large, the exact value of m does not really matter.

Since the value L^{OPT} is known to the algorithm, we may always assume that the instances are rescaled so that L^{OPT} and L^{OPT}/R are convenient numbers (as specified later in the paper).

We call a job *huge* if $p_j \geq L^{\mathrm{OPT}}/R$. Every reasonable algorithm schedules huge jobs on separate machines, because scheduling such a job in any other way wastes the jobs that are assigned to the same machine.

3 The Upper Bound

We analyze our algorithm using an appropriate weight function—a classical technique used for bin packing and related problems.

A weight function $w : \mathcal{R}^+ \to \mathcal{R}^+$ assigns a weight to each job, based on its processing time. The weight of job j is denoted $w_j = w(p_j)$, the total weight of jobs is denoted $W = \sum_j w_j$. Finally, the weight of machine i is defined as

$$W_i = \sum_{j \in S^{-1}(i)} w_j .$$

We illustrate the use of weight functions on a greedy algorithm Init which is known to be $(2-1/m)$-competitive [3]. Assume that $L^{\mathrm{OPT}} = 2-1/m$ (otherwise scale the instance). Fill schedules all jobs greedily on one machine, called an active machine, until it is 1-covered; then it uses a new active machine. As an exception, huge jobs (with $p_j \geq 1$) are always scheduled on a new machine. If no new machine is available, all the remaining jobs are scheduled on the last machine. (This description slightly deviates from [3], however, the behavior is different only when all machines are already 1-covered, so it does not matter for the analysis.)

We define the weight function as $w_j = 2$ for huge jobs (i.e., for jobs with $p_j \geq 1$) and $w_j = p_j$ otherwise. Now every $(2-1/m)$-covered machine has weight $W_i \geq 2-1/m$. Since OPT covers all the machines, it follows that $W \geq 2m-1$. On the other hand, every 1-covered machine generated by Fill has weight $W_i \leq 2$, possibly with the exception of the last machine. Assume that only $m' < m$ machines are 1-covered at the end of the algorithm. Then the 1-covered machines have weight $W_i \leq 2$ each, the last active machine has weight $W_i < 1$, and the remaining machines are empty. Thus the total weight is strictly less than $2m' + 1 \leq 2m - 1$, a contradiction.

To improve upon Fill, we use two active machines in place of a single one. This allows us to avoid the situation when the active machine is almost 1-covered by small jobs and a job of size $1 - \varepsilon$ arrives, causing the final load to be close to 2 in Fill.

Theorem 3.1. *There exists a semi-online algorithm for machine cover which is* $11/6$*-competitive.*

Proof. Without loss of generality, we assume that $L^{\mathrm{OPT}} = 11$ (otherwise scale the instance). We design an algorithm A so that each machine is 6-covered.

The weight function and the total weight. We define the weight function as follows:

$$w_j = \begin{cases} 10 & \text{if } p_j \geq 6 \text{ (huge jobs)} \\ \min(5, p_j) & \text{otherwise} \end{cases}$$

Every 11-covered machine has weight $W_i \geq 10$: It contains either a single huge job, or two jobs of weight 5 (with $p_j \in [5,6)$), or one job of weight 5 and some jobs with $p_j < 5$ and total weight at least 5, or only jobs with $p_j < 5$ of total weight at least 11.

Since OPT has all the m machines 11-covered, the total weight is at least $W \geq 10m$.

The invariants of the algorithm. Our algorithm is designed so that at any time, the total weight of 6-covered machines is at most 10 times their number. In addition, the total weight of jobs on machines that are not 6-covered is strictly less than 10.

Strictly speaking, the invariants may be violated when all the machines but the last one are 6-covered. This final phase of the algorithm needs to be handled separately.

The algorithm. Intuitively, we would like to design the algorithm so that the weight of each machine is at most 10. However, it is not possible to maintain this for each machine. In some cases the algorithm creates pairs of machines with weights at most 9 and 11. The key is to try to create a machine with load (and thus weight) between 2 and 4; upon arrival of a job with $p_j \geq 4$ it is 6-covered with weight at most 9.

The main part of the algorithm is described in Table 1. The algorithm maintains two active machines i and h. All the other machines are at all times either 6-covered or empty.

The leftmost three columns describe four different types of configurations of the algorithm by the conditions on the active machines. The remaining columns describe where a new job is scheduled, depending on its size, and which actions are taken to get back to one of the permitted type of configuration. If a new active machine is requested by the algorithm and none is available, the algorithm enters its final phase described later.

As a rule not included in Table 1, whenever a huge job ($p_j \geq 6$) arrives, it is scheduled on an empty machine, which is 6-covered afterwards. If no empty machine is available, the algorithm enters its final phase described later.

Initially, the active machines are chosen arbitrarily; they are empty and the configuration is INIT. BIG denotes a configuration in which the active machine h actually always contains a single job with $p_j \in [4, 6)$ (as is easily verified by the inspection of Table 1); this guarantees the condition $W_h \leq 5$. GOOD denotes the safe configuration from the intuitive description above with $L_h \in [2, 4)$. Finally, SPEC is a possible successor configuration of GOOD where h is 6-covered with weight at most 9; we still consider this machine active even though no more jobs are scheduled on it. The condition $W_h \leq 9$ in SPEC follows since h contains a single job with $p_j \in [4, 6)$ and possibly some other jobs with total load and weight less than 4.

It is easily verified that when an active machine is closed in BIG or GOOD configurations, its weight is at most 10. When both machines are closed in SPEC, we have $W_h \leq 9$ and $W_i \leq 11$. Also the active 6-covered machine h in SPEC has $W_h \leq 9$. Summarizing, the invariant concerning the weight of 6-covered

Old configuration			New	Action		New
Label	Active machines		job j	Put j on		config.
INIT	$L_h = 0$	$L_i < 2$	$p_j < 2$	i	if $L_i + p_j < 2$	INIT
					otherwise swap $i \leftrightarrow h$	GOOD
			$p_j \in [2, 4)$	h		GOOD
			$p_j \geq 4$	h		BIG
BIG	$L_h \geq 4$ $W_h \leq 5$	$L_i < 2$	$p_j < 2$	i	if $L_i + p_j < 2$	BIG
					otherwise swap $i \leftrightarrow h$	GOOD
			$p_j \geq 2$	h	close h, get a new active machine h	INIT
GOOD	$L_h \in [2, 4)$	$L_i < 6$	$p_j < 4$	i	if $L_i + p_j < 6$	GOOD
					otherwise close i, get a new active machine i	GOOD
			$p_j \geq 4$	h		SPEC
SPEC	$L_h \geq 6$ $W_h \leq 9$	$L_i < 6$	any	i	if $L_i + p_j < 6$	SPEC
					otherwise close i and h, get new machines i and h	INIT

Table 1. The main loop of the 11/6-competitive algorithm

machines is always preserved. Finally, note that in each state, the weight of all the not 6-covered machines is less than 10, as required by the second invariant.

The final phase. It remains to describe and analyze the final phase of the algorithm.

If all the machines are 6-covered upon reaching the final phase, then schedule the remaining jobs on any of the machines.

If a single machine is not 6-covered, schedule all the remaining jobs on this machine. By the invariants, the 6-covered machines have total weight at most $10(m-1)$, the total weight of all jobs is $W \geq 10m$, thus after all jobs are scheduled, the last machine has weight at least 10 and thus it is 6-covered.

If two machines are not 6-covered upon reaching the final phase, then the new machine was requested for a huge job. Schedule this huge job on the machine with the smallest load and all the remaining jobs on the remaining not 6-covered machine. Inspecting the possible configurations, the huge job is scheduled on an active machine with load and weight at most 4. Consequently, similarly to the previous case, after all jobs are scheduled, the last machine has weight at least 6 and thus is 6-covered.

In all the cases, at the end all the machines are 6-covered by the semi-online algorithm, and we conclude that the algorithm is 11/6-competitive.

4 The Lower Bound

Theorem 4.1. *Any deterministic semi-online algorithm for machine cover has competitive ratio at least* 43/24.

Proof. Let ε be such that $1/\varepsilon$ is a large integer, let m be sufficiently large ($m = 44 + 6 \cdot 43/\varepsilon$ works). Without loss of generality, assume that $L^{\mathrm{OPT}} = 43$. Assume for a contradiction that there exists semi-online algorithm A with competitive ratio $43/(24+\varepsilon)$. We construct a counterexample, i.e., an instance for which $L^{\mathrm{A}} < 24 + \varepsilon$.

We formulate the counterexample as a strategy for the adversary, based on how the algorithm A scheduled the jobs so far. The adversary wins the game when it is possible to modify the schedule produced by the algorithm A to get some (possibly suboptimal) schedule which has more 43-covered machines than is the number of $(24+\varepsilon)$-covered machines of A. Strictly speaking, after this the adversary continues with jobs of size 43 until all the machines are covered.

Finally, we can assume without loss of generality that the algorithm A never schedules a job on any $(24+\varepsilon)$-covered machine. (A new machine is always available as m is large.)

Throughout the proof, the content of a machine is written in braces as numbers denoting jobs of those sizes In addition, a number in square brackets denotes a set of jobs with this total size. Thus, for example, $\{9, 9, 10, [15]\}$ denotes a machine with total load 43 which contains two jobs of size 9, one job of size 10 and some other jobs.

Phase 0 The instance starts with a sequence of $2 \cdot 43/\varepsilon$ jobs of size 24. The optimum can create $43/\varepsilon$ of 43-covered machines, each containing two of the jobs. Thus at the end of the phase, the algorithm A also has $43/\varepsilon$ machines with two jobs, i.e., $\{24,24\}$, as otherwise the adversary wins.

Phase 1 The goal of phase 1 is to make the algorithm A to create $4 \cdot 43/\varepsilon$ machines of form $\{5, 15, 24\}$ or alternatively $2 \cdot 43$ machines $\{9, 15, 19\}$. Table 2 shows the strategy of the adversary for this phase. The table shows only nonempty machines that are not $(24+\varepsilon)$-covered, or are newly covered (marked by a star). The first column describes possible configurations of the schedule of A in this phase. The second column gives the job submitted by the adversary for each configuration, and the last column describes all the possible configurations of A after the new job is scheduled.

It is easy to verify that all the machines $(24+\varepsilon)$-covered by the algorithm A so far are also 43-covered.

The adversary stops in the situations marked in the table as winning. If the configuration is $\{5, 15\}, \{24\}$ or $\{9, 15\}, \{5, 19\}$ or $\{9, 15\}, \{5\}, \{19\}$ then the load on the uncovered machines is more than 43, and the adversary wins by reassigning these jobs on a single 43-covered machine (all the other machines stay as in the schedule of A). In configurations $\{15\}, \{9, 5\}$ and $\{15\}, \{9\}, \{5\}$ the adversary submits two additional jobs, one of size 5 and one of size 4. The algorithm A cannot cover another machine, but the adversary can convert the schedule using one $\{24, 24\}$ machine to a schedule with two machines $\{24, 15, 4\}$ and $\{24, 9, 5, 5\}$, so the adversary wins again.

If no such situation is encountered, then the adversary waits until the algorithm A covers $4 \cdot 43/\varepsilon$ machines by jobs $\{5, 15, 24\}$ or $2 \cdot 43$ machines by jobs

Old configuration	New job	Possible new configurations
$\emptyset$	5	$\{5\}$
$\{5\}$	15	$\{5, 15\}$ $\{5\}, \{15\}$
$\{5, 15\}$	24	$\{5, 15, 24\}^*$, $\emptyset$ $\{5, 15\}, \{24\}$ – the adversary wins
$\{5\}, \{15\}$	9	$\{5\}, \{9, 15\}$ $\{5, 9\}, \{15\}$ – the adversary wins $\{5\}, \{9\}, \{15\}$ – the adversary wins
$\{5\}, \{9, 15\}$	19	$\{9, 15, 19\}^*, \{5\}$ $\{9, 15\}, \{5, 19\}$ – the adversary wins $\{9, 15\}, \{5\}, \{19\}$ – the adversary wins

Table 2. The strategy of the adversary in phase 1. The machines marked by star are newly covered (and thus removed from the configuration).

$\{9, 15, 19\}$, and then continues with phase 2. Note that in the final configuration, either there is no non-empty (not covered) machine, or there is one machine $\{5\}$.

Phase 2 During this phase, let i_1 and i_2 be the indices of the two uncovered machines with the largest loads. I.e., L_{i_1} is the maximal load of an uncovered machine.

The phase proceeds in $43/\varepsilon$ rounds. The adversary maintains a rearranged schedule, starting with the schedule of the algorithm A after phase 1. After each round, if the adversary has not yet won, it rearranges some of the machines from the previous phases and the new jobs so that it has as many 43-covered machines as A has $(24+\varepsilon)$-covered. In addition, in each such rearrangement the adversary saves at least one job of size ε.

At the beginning of each round, we have some not covered machines with loads at most 14, containing jobs of size ε and possibly one job of size 5. The not covered machines in the rearranged schedule of the adversary may contain jobs different from the jobs on the machines of A, but the loads are the same.

Now we describe one round of phase 2. The adversary submits jobs of size ε until $L_{i_1} = 24$. If $L_{i_2} \geq 14$, the adversary converts the schedule using one machine $\{24, 24\}$ to create two machines $\{24, [19]\}$ and wins. Otherwise $L_{i_2} \leq 14 - \varepsilon$ and the adversary submits a job of size $X = 24 - L_{i_2} \geq 10 + \varepsilon$. The algorithm A has to create a machine $\{X, [24]\}$, as otherwise the adversary uses the jobs from not covered machines to create one 43-covered machine and wins. Finally, the adversary submits jobs of size ε until $L_{i_1} \geq 5$.

Now we describe how the machines are rearranged. First, if the newly covered machine $\{X, [24]\}$ contains the job of size 5, then this job is exchanged with $5/\varepsilon$ jobs of size ε from machine i_1. At this point, the machine $\{X, [24]\}$ contains only X and jobs of size ε. Next, using this machine and some machines from the previous phases, the adversary uses one of following conversions (see Figure 1 for an illustration of the conversion (1)):

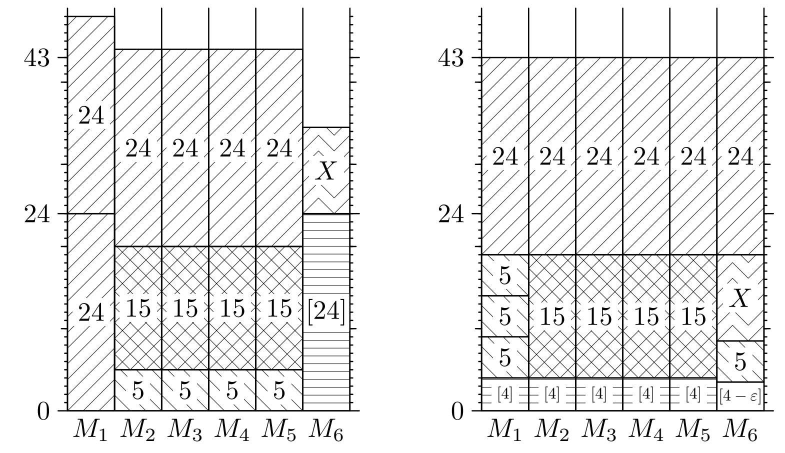

Fig. 1. Conversion (1) of the schedule of the semi-online algorithm (left) to a better schedule (right) with a saved job of size ε. Machine M_1 is from phase 0, machines $M_2, \dots M_5$ from phase 1, and machine M_6 is created in phase 2.

$$\begin{aligned}&\{24,24\}, 4\times\{5,15,24\}, \{X,[24]\}\\ &\qquad\to \{24,5,5,5,[4]\}, 4\times\{24,15,[4]\}, \{24,X,5,[4-\varepsilon]\}, \varepsilon\end{aligned} \tag{1}$$

$$\begin{aligned}&2\times\{24,24\}, 2\times\{9,15,19\}, \{X,[24]\}\\ &\qquad\to 2\times\{24,19\}, 2\times\{24,15,[4]\}, \{9,9,X,[15]\}, [1]\end{aligned} \tag{2}$$

After this conversion, the number 43-covered machines in the schedule of the adversary is equal to the number of $(24+\varepsilon)$-covered machines in the schedule of A. So the adversary may continue with another round of the phase 2.

The number of machines covered in phases 0 and 1 guarantees that $43/\varepsilon$ conversions (1) or 43 conversions (2) are always possible in phase 2. When phase 2 is complete, the adversary saved at least $43/\varepsilon$ jobs of size ε. Now the adversary uses these jobs to create a new 43-covered machine and wins.

As the adversary eventually always wins, we conclude that there is no 43/24-competitive algorithm.

Acknowledgments.

We are grateful to anonymous referees for many useful comments. T. Ebenlendr and J. Sgall were partially supported by Institutional Research Plan No. AV0Z10190503, by Inst. for Theor. Comp. Sci., Prague (project 1M0545 of MŠMT ČR), and grant 201/05/0124 of GA ČR.

References

1. N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *J. Sched.*, 1:55–66, 1998.
2. Y. Azar and L. Epstein. Approximation schemes for covering and scheduling on related machines. In *APPROX*, volume 1444 of *Lecture Notes in Comput. Sci.*, pages 39–47. Springer, 1998.
3. Y. Azar and L. Epstein. On-line machine covering. *J. Sched.*, 1:67–77, 1998.
4. L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica*, 39:43–57, 2004.
5. D. Friesen and B. Deuermeyer. Analysis of greedy solutions for a replacement part sequencing problem. *Math. Oper. Res.*, 6:74–87, 1981.
6. G. J. Woeginger. A polynomial time approximation scheme for maximizing the minimum machine completion time. *Oper. Res. Lett.*, 20:149–154, 1997.