# Approximation schemes for covering and scheduling on related machines

Yossi Azar[1],Leah Epstein[2]

[1] Dept. of Computer Science, Tel-Aviv University. ***
[2] Dept. of Computer Science, Tel-Aviv University. †

**Abstract.** We consider the problem of assigning a set of jobs to $m$ parallel related machines so as to maximize the minimum load over the machines. This situation corresponds to a case that a system which consists of the $m$ machines is alive (i.e. productive) only when all machines are alive, and the system should be maintained alive as long as possible. The above problem is called related machines covering problem and is different from the related machines scheduling problem in which the goal is to minimize the maximum load. Our main result is a polynomial approximation scheme for this covering problem. To the best of our knowledge the previous best approximation algorithm has a performance ratio of 2. Also, an approximation scheme for the special case of identical machines was given by [14].
Some of our techniques are built on ideas of Hochbaum and Shmoys [12]. They provided an approximation scheme for the well known related machines scheduling. In fact, our algorithm can be adapted to provide a simpler approximation scheme for the related machines scheduling as well.

## 1 Introduction

We consider the problem of assigning a set of jobs to $m$ parallel related machines so as to maximize the minimum load over the machines. This situation is motivated by the following scenario. A system which consists of $m$ related machines is alive (i.e. productive) only when all machines are alive. The duration that a machine is alive is proportional to the total size of the resources (e.g. tanks of fuel) allocated to it. The goal is to keep the system alive as long as possible using a set of various sizes resources. The above problem has applications also in sequencing of maintenance actions for modular gas turbine aircraft engines [8]. To conform with the standard scheduling terminology we view the resources as jobs. Thus, jobs are assigned to machines so as to maximize the minimum load. In the related machines case each machine has its own speed

*** E-Mail: azar@math.tau.ac.il. Research supported in part by the Israel Science Foundation and by the United States-Israel Binational Science Foundation (BSF).
† E-Mail: lea@math.tau.ac.il.

(of the engine that operates on fuel). The identical machines case is a special case where all speeds of machines are identical. We refer to these problems as machines covering problems. Note that the classical scheduling/load-balancing problems [10, 11, 12, 13] seem strongly related to the covering problems. However, scheduling/load-balancing are packing problems and hence their goal is to minimize the makespan (i.e. to minimize the maximum load over all machines) where in the covering problems the goal is to maximize the minimum.

**Our results:** Our main result is polynomial approximation scheme for the covering problem in the related machines case. That is for any $\varepsilon > 0$ there is a polynomial time algorithm $A_\varepsilon$ that approximates the optimal solution up to a factor of $1+\varepsilon$. In fact, since the problem is strongly NP hard no fully polynomial approximation scheme exists unless P=NP. Some of our techniques are built on ideas of Hochbaum and Shmoys [12]. They provided an approximation scheme for the well known related machines scheduling. In fact, our algorithm can be adapted to provide a simpler approximation scheme for the related machines scheduling as well.

**Known results:** The problem of maximizing the load of the least loaded machine, (i.e the machines covering problem) is known to be NP-complete in the strong sense already for identical machines [9]. For the identical machines case Deuermeyer, Friesen and Langston [7] studied the LPT-heuristic. The LPT-heuristic orders the jobs by non increasing weights and assigns each job to the least loaded machine at the current moment. It is shown in [7] that the approximation ratio of this heuristic is at most $\frac{4}{3}$. The tight ratio $\frac{4m-2}{3m-1}$ is given by Csirik, Kellerer and Woeginger [5]. Finally, Woeginger [14] designed a polynomial time approximation scheme for the identical machines covering problem.

The history for the *related* machines covering problem is much shorter. The only result which is known for the related machines case is the $2+\varepsilon$ approximation algorithm follows from [4]. The above paper also contains results for the on-line machines covering problems.

**Definitions:** We give a formal definition of the problems discussed above. Consider a set of $m$ identical machines and a set of jobs. Machine $i$ has a speed $v_i$ and a job $j$ has a weight $w_j$. The load of a machine $i$ is the sum of the weights of the jobs assigned to it normalized by the speed. That is, $\ell_i = \sum_{j\in J_i} \frac{w_j}{v_i}$ where $J_i$ is the of jobs assigned to machine $i$. (The identical machines case is the special case where all $v_i$ are equal.) The goal in the machines covering problems is to assign the jobs to the machines so as to maximize the minimum load over the machines. This is in contrast to the scheduling/load-balancing problems where the goal there is to minimize the maximum load. Note that these covering problems are also different from the bin covering problems [1, 2, 3, 6] where the goal is to maximize the number of covered bins, i.e. bins of load of at least 1.

## 2 Approximation scheme for machine covering

We use a standard binary search technique to search for the value of the optimal cover. By this we reduce the approximation algorithm to the following approximate decision problem. Given a value $T$ for the algorithm, the decision procedure outputs an assignment of value at least $(1-\varepsilon)T$, or answers that there does not exist an assignment of value at least $T$. We start the binary search with an initial estimation of the value of the optimal cover using the $(2+\varepsilon)$-approximation algorithm given in [4]. Clearly, the overall complexity of the approximation algorithm is just $O(\log 1/\varepsilon)$ times the complexity of the decision procedure (the initial estimation algorithm is fast).

We note that the decision procedure is equivalent to decide if one can fill the $m$ bins such that bin $i$ is filled by at least $(1-\varepsilon)Tv_i$. We scale the sizes of bins and the weights of jobs, so that the size of the smallest bin is 1. Now we have a set of $n$ jobs, and a set of $m$ bins of sizes $s_1, s_2, ..., s_m$ where $1 = s_1 \leq s_2 \leq ... \leq s_m$.

**Bin ranges:** We partition the bins according to their sizes into sets $B_r$, where the bin range set $B_r$ is the set of all bins of size $2^r \leq s_j < 2^{r+1}$. Let $R = \{r | B_r \neq \phi\}$, clearly $|R| \leq m$. We choose $\varepsilon_0$ to be a value such that $\frac{\varepsilon}{16} \leq \varepsilon_0 \leq \frac{\varepsilon}{8}$ and $\frac{1}{\varepsilon_0}$ is an integer. We denote $\varepsilon_r = 2^r \varepsilon_0$, $\delta_0 = \varepsilon_0^2$ and $\delta_r = 2^r \delta_0$. For each bin range $B_r$ the jobs are partitioned into three sets.

- Big jobs: jobs of weight more than $2^{r+1}$.
- Medium jobs: jobs of weight $w_j$: $\varepsilon_r < w_j \leq 2^{r+1}$.
- Small jobs: jobs of weight at most $\varepsilon_r$.

**Jobs vectors:** For each $B_r$ we can approximate a set of jobs by a jobs vector $(y, n_1, n_2, \ldots, n_l, W)$ where $y$ is the number of big jobs, $n_k$ is the number of (medium) jobs whose size is between $t_{k-1}$ and $t_k$ where $t_k = \varepsilon_r + k\delta_r$ and $W$ is the total weight of small jobs in the set. Clearly $l$, the number of types of medium jobs, is at most $\frac{2}{\delta_0} - \frac{1}{\varepsilon_0} \leq \frac{2}{\delta_0}$. We refer to the values of $t_k$ as rounded weights. Note that the jobs vector for a given set of jobs depends on the bin range.

**Cover vectors:** Let $B_r$ be the bin range of bin $j$. A cover vector for bin $j$ has the same form as the jobs vector except the last coordinate which is an integer that corresponds to the weight of small jobs normalized by $\varepsilon_r$. A vector $(y, n_1, n_2, \ldots, n_l, q)$ is a cover vector for bin $j$ if

$$2^{r+1}y + \sum_{k=1}^{l} n_k t_k + q\varepsilon_r \geq s_j - \varepsilon_r = s_j(1-\varepsilon_0) \ ,$$

i.e., the sum of the rounded weights of the jobs in the vector is at least $1-\varepsilon_0$ fraction of $s_j$.

Let $T'_j$ be the set of cover vectors for a bin $j$. Let $T_j$ be the set of minimal cover vectors with respect to inclusion i.e. a cover vector $u$ is in $T_j$ if for any other cover vector for bin $j$, $u'$ the vector $u - u'$ has at least one negative coordinate. Since the minimum weight of a job is $\varepsilon_0 2^r$ and the size of the bin is at most $2^{r+1}$ then any minimal cover vector consists of at most $\frac{2}{\varepsilon_0}$ jobs (sum of coordinates). Clearly, we may use only minimal vector covers since any cover can be transformed to a minimal one by omitting some jobs.

**The layer graph:** We use a layer graph where each node of the graph is state vector which is in a form of a cover vector. We order the bins in non decreasing size order. The layers of the graph are partitioned into phases. There are $|R|$ phases, a phase for each $r \in R$. Let $b_r = |B_r|$ for $r \in R$ and $b_r = 0$ otherwise. Phase $r$ consists of $b_r + 1$ layers $L_{r,0}, ..., L_{r,b_r}$. The nodes of $L_{r,i}$ are all admissible state vectors of $B_r$. We put an edge between a node $x'$ in $L_{r,i-1}$, and a node $x$ in $L_{r,i}$ if the difference $u' = x' - x$ is a minimal cover vector of bin $j$.

Layer $L_{r,i}$ corresponds to jobs that remained after the first $j = \sum_{t=0}^{r-1} b_t + i$ bins were covered. More specifically, if there is a path from $L_{0,0}$ to a state vector in $L_{r,i}$ then there is a cover of the first $j$ bins, where bin $k \leq j$ is covered with $s_k(1-\varepsilon_0)$, such that the remaining jobs has jobs vector which is identical to the state vector except of the last coordinate. The last coordinate of the jobs vector, $W$, satisfies $q\varepsilon_r \leq (W + 2\varepsilon_r)(1+\varepsilon_0)$ where $q$ is the last coordinate of the state vector. Moreover, if there is a cover of the first $j$ bins, bin $k \leq j$ with $s_k$ such that the remaining jobs set defines some jobs vector then there a path from the first layer to the node in layer $L_{r,i}$ whose state vector is identical to the jobs vector except of the last coordinate. The last coordinate of the state vector, $q$, satisfies $W \leq q\varepsilon_r$ where $W$ is the last coordinate of the jobs vector.

**The translating edges between phases:** The edges between phases "translate" each state vector into a state vector of the next phase. These edges are not used to cover bins, but to move from one phase to another. There is only one outgoing edge from each node in a last layer of any phase (except the last one which has no outgoing edges). More specifically, for each phase $r$, any node in $L_{r,b_r}$ translates by an edge into a node in $L_{r',0}$ where $r' > r$ is the index of the next phase.

We consider a state vector in $L_{r,b_r}$ (an **input vector**). We construct a corresponding state vector in layer $L_{r',0}$ (an **output vector**) which results in an edge between them. We start with an empty output state vector. We scan the input state vector. To build the output vector we need to know how jobs change their status. Since the bins become larger, the small jobs remain small. Medium jobs may either become small, or stay medium. Big jobs can stay big, or become medium or small.

First we consider the number of big items $y$ in the input vector. Note that all big jobs could be used to cover previous bins. Thus we may assume that the smallest big jobs were used, and the $y$ big jobs that remained are the largest $y$ jobs in the original set. Let $y_1$ be the number of big jobs in the input vector that are also big in $B_{r'}$, $y_2$ the number of jobs that are medium in $B_{r'}$ and $y_3$ the number of jobs that are small in $B_{r'}$.

A medium job in phase $B_r$ becomes small in $B_{r'}$ if its rounded weight $t_j$ is at most $\varepsilon_{r'}$. In phase $B_r$ the rounded weight of a job of weight $\varepsilon_{r'}$ is exactly $\varepsilon_{r'}$ since $(2^{r'-r}-1)/\varepsilon_0$ is integer, and $\varepsilon_{r'} = 2^{r'}\varepsilon_0 = 2^r\varepsilon_0 + ((2^{r'-r}-1)/\varepsilon_0 \cdot 2^r\varepsilon_0^2)$. Thus all medium jobs with $k \leq (2^{r'-r}-1)/\varepsilon_0$ become small, and all other medium jobs remain medium.

**The coordinates of the output vector:** The big job coordinate in the output vector would be $y_1$, since no medium or small jobs could become big.

Now we deal with all the jobs which became small in the output, i.e. the $y_3$

big jobs together with the medium input jobs that became small output jobs and all the small jobs which must remain small in the output. To build the component of small jobs we re-estimate the total weight of small jobs. Since small jobs remain small, we initialize $W' = q\varepsilon_r$ where $q$ is the integer value of the small jobs in the input node. We add to $W'$ the total sum of the rounded jobs that were medium in $B_r$ and become small in $B_{r'}$ (their rounded weight in $B_r$), and also the original weight of the big jobs in $B_r$ that become small in $B_{r'}$. The new component $q'$ of small jobs is $\lceil W'/\varepsilon_{r'} \rceil$.

Next we consider all jobs that are medium in $B_{r'}$. There were $y_2$ big jobs in $B_r$ that become medium in $B_{r'}$. For every such job we round its weight according to $B_{r'}$ and add one to the coordinate of its corresponding type in $B_{r'}$. What remains to consider is the coordinates of jobs that are medium for both $B_r$ and $B_{r'}$. We claim that all jobs that are rounded to one type $k$ in $B_r$ cannot be rounded to different types $B_{r'}$. Thus we add the type $k$ coordinate of the input vector to the corresponding coordinate of the output vector. To prove the claim we note that a job of type $k$ satisfies $2^r\varepsilon_0 + (k-1)2^r\varepsilon_0^2 < w_j \le 2^r\varepsilon_0 + k2^r\varepsilon_0^2$. Let $l = r' - r$, in terms of $\varepsilon_{r'}$ and $\delta_{r'}$ we get that

$$\varepsilon_{r'} + \delta_{r'} \left(\frac{1}{2^l}(\frac{1-2^l}{\varepsilon_0} + k - 1)\right) < w_j \le \varepsilon_{r'} + \delta_{r'} \left(\frac{1}{2^l}(\frac{1-2^l}{\varepsilon_0} + k)\right) .$$

Since $\frac{1-2^l}{\varepsilon_0} + k - 1$ and $\frac{1-2^l}{\varepsilon_0} + k$ are integers, then the interval $(\frac{1}{2^l}(\frac{1-2^l}{\varepsilon_0} + k - 1), \frac{1}{2^l}(\frac{1-2^l}{\varepsilon_0} + k))$ does not contain an integer and thus all these jobs are rounded to the type $\varepsilon_{r'} + \delta_{r'}\lceil \frac{1}{2^l}(\frac{1-2^l}{\varepsilon_0} + k)\rceil$.

After we built the graph, we look for a path between a node of the first layer in the first phase which corresponds to the whole set of jobs, and any node in the last layer of the last phase (which means that we managed to cover all bins with the set of jobs, maybe some jobs were not used but it is possible to add them to any bin). If such path exists, the procedure answers "yes", and otherwise "no".

## 3 Analysis of the algorithm

In this section we prove the correctness of our algorithm and compute its complexity.

**Theorem 1.** *If a feasible cover of value $T$ exists, then the procedure outputs a path that corresponds to a cover of at least $(1-\varepsilon)T$. Otherwise, the procedure answers "no".*

*Proof.* We need to show that any cover of value at least $T$ can be transformed into a path in the graph, and that a path in the graph can be transformed into a cover of at least $(1-\varepsilon)T$.

Assume a cover of value $T$. We first transform it to a cover of a type that our algorithm is searching for. We assume that the cover of any bin is minimal, otherwise we just remove jobs. Next we scan the bins by their order in the layered graph (small to large). If the $j$'th bin was covered by a single job, we change the cover to get another feasible cover in the following way. We consider the smallest job that is at least as large as the $j$'th bin, and is not used to cover a bin with a smaller index. We change the places of the two big jobs and continue the scanning. The modified cover after each change is still feasible since bin $j$ is still covered and weight of the jobs on the larger bin may only increase. We use the final cover to build a path in the graph. We show how to add a single edge to a path in each step. Since there is only one outgoing edge from each node of the last layer in each phase, we only need to show how to add edges between consecutive layers inside the phases, each such edge corresponds to some bin. We also assume inductively the following invariant. The weight of the small jobs in any state vector is at least the weight of the small jobs that has not been yet used in the cover (the cover for bins we already considered).

Consider the $j$'th bin. If the bin is covered with a big job then we choose an edge that corresponds to one big job. Otherwise the bin is covered by medium jobs and small jobs. We compute the following state vector. Each medium job adds one to the appropriate coordinate of the vector. Note that rounded weight of a medium job (as interpreted in the state vector) is at least as large as its weight. We also need to provide one coordinate for the small jobs used in covering bin $j$. For that we divide the total weight of the small jobs in the cover of that bin by $\varepsilon_r$ where $r$ is the index of the phase and take the floor. The sum of the rounded weights of the jobs of the vector we built is at least $s_j - \varepsilon_r$, and thus it is a cover vector, i.e., an edge in the $j$'th layer of the graph. the invariant on the small jobs remains true since the weight of the small jobs of the cover vector is less than that in the cover.

Now we show that a full path in the graph is changed into a cover of value $(1-\varepsilon)T$. We build the cover starting from the smallest bin (the first edge of the path). We show how to build a cover for one bin using a cover vector. In the case that the cover vector of the edge corresponds to one big job, we add the smallest big job available to the cover. We replace a medium job of rounded weight $\varepsilon_r + k\delta_r$ by a job of weight in the interval $(\varepsilon_r + (k-1)\delta_r, \varepsilon_r + k\delta_r]$. Such a job must exist according to the way that the graph was built. We replace the small jobs coordinate in the following way: Let $j_1, ..., j_z$ where $w_{j_1} \leq ... \leq w_{j_z}$ be the subset of the small jobs that were not used to cover any of the bins that were already covered. Let $z_1$ be the index $1 \leq z_1 \leq z$ such that

$$(q'-3)\varepsilon_r < \sum_{1 \leq i \leq z_1} w_{j_i} \leq (q'-2)\varepsilon_r$$

(where $q'$ is the small jobs coordinate of the cover vector). We add the jobs $j_1, ..., j_{z_1}$ to the cover. We prove the following Lemma in order to show that such an index exists.

**Lemma 2.** *Consider the node in phase $r$ on the path up to which we built the cover. Denote by $W_1$ the total rounded weight of small jobs that were not used to cover any of the bins that are already covered. (The rounded weight of a small job is its rounded weight when it was last medium, and its real weight if it never* *was medium). The small jobs coordinate $q$ of the state vector of the node satisfies the equation*

$$q\varepsilon_r \leq W_1 + 2\varepsilon_r$$

*Proof.* First we show the correctness of the lemma only for the first layer of each phase. We show the lemma by induction on the number of phase $r$. In the beginning of each phase there is a rounding process in which the total rounded weight of small jobs is divided by $\varepsilon_r$, and the result is rounded up. Thus in phase 0 since we round up the number $W'/\varepsilon_0$, we added at most $\varepsilon_0$ to the total rounded weight of the jobs. In phase $r$ we might add by rounding extra $\varepsilon_r$. Since $\sum_{r'<r} \varepsilon_{r'} < \varepsilon_r$ we conclude that the weights of the small jobs of a state vector in the first layer of a phase exceeds the total rounded weight of the small jobs available by at most $2\varepsilon_r$ and thus $q\varepsilon_r \leq W_1 + 2\varepsilon_r$. The lemma holds for nodes inside phases too, since each time we reduce $q$, by some number $q_1$, $W_1$ is reduced by at most $(q_1 - 2)\varepsilon$. This completes the proof of the lemma.

In a cover vector the small jobs coordinate $q'$ is bounded above by the small jobs coordinate of the node for which this edge is outgoing. Thus $q'\varepsilon_r \leq q\varepsilon_r \leq W_1 + 2\varepsilon_r$ and thus $W_1 \geq (q' - 2)\varepsilon_r$. Since the weight of each job is at most $\varepsilon_r$, $z_1$ exists.

Now we show that this gives a cover of bin $j$. Let us calculate the ratio between the rounded weight and the real weight of jobs we assigned to this bin. Since we rounded only medium jobs, the loss in the weight of the job is at most $\delta_r$, while its weight is at least $\varepsilon_r$. Thus the ratio is at most $(w_k + \delta_r)/w_k \leq 1 + \delta_r/\varepsilon_r \leq 1 + \varepsilon_0$. Thus the total weight of the jobs assigned to bin $j$ is at least

$$(s_j - 4 \cdot \varepsilon_r)/(1+\varepsilon_0) \geq (s_j - 4 \cdot 2^r \varepsilon_0)(1 - \varepsilon_0)$$
$$\geq s_j(1 - 4\varepsilon_0)(1 - \varepsilon_0) \geq s_j(1 - 5\varepsilon_0) \geq s_j(1 - \varepsilon)$$

where the second inequality follows from the fact that $s_j \geq 2^r$.

**Complexity:** The decision procedure consists of two parts.

1. Building the graph.
   The number of nodes in each layer of the graph is at most $N = 2(n + 2)n^{2/\varepsilon_0^2+1}$ since each of the coordinates of the first and the second parts of state vector does not exceed $n$, and the third one does not exceed $(n+2)(1+\varepsilon_0) < 2(n+2)$. Consider the a translation edges between layers of different phases. Recall that there is only one edge from each node. Now, consider the edges between layers that correspond to a bin. The number of edges from each node is at most $M = (2/\varepsilon_0^2 + 3)^{2/\varepsilon_0}$ (Since each of the $2/\varepsilon_0$ jobs can be one of $2/\varepsilon_0^2 + 2$ jobs types, or not to exist at all in the cover). Thus there are at most $mNM + mN$ edges in the graph. To build the edges inside phases we need to check at most $M$ possibilities for each node inside a phase. It takes $O(\frac{1}{\varepsilon})$ to check one such possibility, i.e. that it corresponds to a minimal cover. Building one edge between phases takes $O(n)$ but there is only one outgoing edge from each node between phases. The total complexity of building the graph is $O(mNM/\varepsilon + mNn)$.

2. Running an algorithm to find a path, and translating it into a cover if it succeeds.
   The algorithm of finding a path is linear in the number of edges. The complexity of building the cover is negligible.

The total complexity of the decision procedure is $O(c_\varepsilon n^{O(\frac{1}{\varepsilon})})$ where $c_\varepsilon$ is a constant which depends only on $\varepsilon$. The complexity of the algorithm is clearly the complexity of the decision procedure times $O(\log 1/\varepsilon)$.

## 4 Approximation scheme for scheduling

It is easy to adapt the algorithm for machines covering and get a simpler algorithm for related machines scheduling. Here the big jobs cannot not be used for packing in bins and thus the algorithm becomes simpler. Of course, there are several straight forward changes such as rounding down the jobs instead of rounding them up and considering maximum packings instead of minimum covers. We omit the details.

## References


1. N. Alon, J. Csirik, S. V. Sevastianov, A. P. A. Vestjens, and G. J. Woeginger. On-line and off-line approximation algorithms for vector covering problems. In *Proc. 4th European Symposium on Algorithms*, LNCS, pages 406–418. Springer, 1996.
2. S.F. Assmann. Problems in discrete applied mathematics. Technical report, Doctoral Dissertation, Mathematics Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1983.
3. S.F. Assmann, D.S. Johnson, D.J. Kleitman, and J.Y.-T. Leung. On a dual version of the one-dimensional bin packing problem. *J. Algorithms*, 5:502–525, 1984.
4. Y. Azar and L. Epstein. On-line machine covering. In *5th Annual European Symposium on Algorithms*, pages 23–36, 1997.
5. J. Csirik, H. Kellerer, and G. Woeginger. The exact lpt-bound for maximizing the minimum completion time. *Operations Research Letters*, 11:281–287, 1992.
6. J. Csirik and V. Totik. On-line algorithms for a dual version of bin packing. *Discr. Appl. Math.*, 21:163–167, 1988.
7. B. Deuermeyer, D. Friesen, and M. Langston. Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM J. Discrete Methods*, 3:190–196, 1982.
8. D. Friesen and B. Deuermeyer. Analysis of greedy solutions for a replacement part sequencing problem. *Math. Oper. Res.*, 6:74–87, 1981.
9. M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, San Francisco, 1979.
10. R.L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
11. R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math*, 17:263–269, 1969.
12. D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

13. D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *J. Assoc. Comput. Mach.*, 34(1):144–162, January 1987.
14. G. Woeginger. A polynomial time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20:149–154, 1997.