The Open Shop Scheduling Problem

Gerhard J. Woeginger
Department of Computer Science, RWTH Aachen, D-52074 Aachen, Germany
woeginger@algo.rwth-aachen.de

Abstract

We discuss the computational complexity, the approximability, the algorithmics and the combinatorics of the open shop scheduling problem. We summarize the most important results from the literature and explain their main ideas, we sketch the most beautiful proofs, and we also list a number of open problems.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms, Mathematics of computing → Discrete optimization

Keywords and phrases Algorithms, Complexity, Scheduling, Approximation

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.4

Category Invited Talk

1 Blacksmiths and horseshoes

"There are four blacksmiths working together. One of them has specialized in putting horseshoes on the left front leg of a horse, while the other three have specialized respectively in putting horseshoes on the left hind leg, the right front leg, and the right hind leg. If the work on one horseshoe takes five minutes, what is the minimum amount of time needed to put twenty-eight horseshoes on seven horses? (Note that a horse cannot stand on two legs.)"

As each blacksmith has to handle 7 horseshoes, he needs at least 35 minutes of working time. The following picture with horses A, B, C, D, E, F, G and blacksmiths $1, 2, 3, 4$ shows a schedule that meets this lower bound of 35 minutes. Note that each horse receives its four horseshoes during four different time slots (so that it never has to stand on two legs), and note that during each five minute time slot each blacksmith works for exactly five non-interrupted minutes on a single horse.

	minute 00–05	minute 05–10	minute 10–15	minute 15–20	minute 20–25	minute 25-30	minute 30-35
Blacksmith 1	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Blacksmith 2	**B**	**C**	**D**	**G**	**F**	**E**	**A**
Blacksmith 3	**C**	**D**	**G**	**E**	**A**	**B**	**F**
Blacksmith 4	**D**	**A**	**F**	**B**	**C**	**G**	**E**

2 Problem statement and some definitions

An instance of the *open shop* scheduling problem consists of m machines $M_1, \ldots, M_m$ and n jobs $J_1, \ldots, J_n$. (Throughout, machines will be indexed by i and jobs will be indexed by j.) Each job J_j consists of m independent operations $O_{i,j}$ with $i = 1, \ldots, m$. The operation $O_{i,j}$ of job J_j has to be processed on machine M_i, which takes $p_{i,j}$ uninterrupted time units. For

© Gerhard J. Woeginger;
licensed under Creative Commons License CC-BY
35th Symposium on Theoretical Aspects of Computer Science (STACS 2018).
Editors: Rolf Niedermeier and Brigitte Vallée; Article No. 4; pp. 4:1–4:12
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

every job, the order in which its operations have to be processed is *not fixed* in advance but may be chosen arbitrarily by the scheduler; we stress that different jobs may receive different processing orders.

A *schedule* assigns every operation $O_{i,j}$ to a time interval of length $p_{i,j}$, so that no job is simultaneously processed on two different machines and so that no machine simultaneously processes two different jobs. The *makespan* $C_{\max}$ of a schedule is the largest job completion time. The optimal makespan is usually denoted by $C^*_{\max}$. In the standard scheduling classification scheme of Lawler, Lenstra, Rinnooy Kan & Shmoys [12], this optimization problem is denoted by $O||C_{\max}$ (if the number m of machines is given as part of the input) and by $Om||C_{\max}$ (if the number m of machines is a fixed constant number).

In the *"Blacksmiths and horseshoes"* puzzle, the four blacksmiths are four machines $M_1, M_2, M_, M_4$. Each horse forms a job, and its four legs are the four operations of that job. All operations $O_{i,j}$ have length $p_{i,j} = 5$.

Here are some more notations. The length of the longest operation in an instance is denoted $o_{\max} = \max_{i,j} p_{i,j}$. The overall processing time of job J_j will be denoted $p_j = \sum_{i=1}^{m} p_{i,j}$. The overall processing time assigned to machine M_i is called the *load* $L_i = \sum_{j=1}^{n} p_{i,j}$ of the machine. The maximum job processing time is denoted $p_{\max} = \max_j p_j$ and the maximum machine load is denoted $L_{\max} = \max_i L_j$. As no job can be simultaneously processed on two different machines the makespan of any schedule satisfies $C_{\max} \geq p_{\max}$, and as no machine can simultaneously processes two different jobs the makespan satisfies $C_{\max} \geq L_{\max}$. This yields the following lower bound, which will be important throughout the paper:

$$C^*_{\max} \;\geq\; \beta^* \;:=\; \max\{L_{\max}, p_{\max}\} \tag{1}$$

We mention in passing that there are two other important shop scheduling problems that are closely related to the open shop problem: In a *flow shop*, every job must pass the machines in the same ordering $M_1, \ldots, M_m$. In a *job shop*, the ordering of the operations is fixed a priori for every job, and different jobs may have different orderings of operations. These variants will not be further discussed in the rest of this paper.

3 Computational complexity

Gonzalez & Sahni [10] prove that the open shop on $m = 2$ machines allows a very simple polynomial time solution: There always exists a schedule whose makespan equals the lower bound β^* in (1), and this schedule can be found in linear time.

▶ **Theorem 1** (Gonzalez & Sahni [10]). *Problem $O2||C_{\max}$ is solvable in polynomial time.*

The algorithm in Theorem 1 is not hard to find (there are many possible approaches), and we leave it as a puzzle for the reader. A more general problem variant considers the completion time C_1 of the last operation on machine M_1 and the completion time C_2 of the last operation on M_2, and asks for a schedule that minimizes some objective function $f(C_1, C_2)$ of the two machine completion times. Based on extensive case distinctions, Shaklevich & Strusevich [21] develop linear time algorithms for this variant, if the function $f(\cdot,\cdot)$ is non-decreasing in both arguments. Van den Akker, Hoogeveen & Woeginger [23] provide a simpler proof of the same result. Sahni & Cho [14] prove strong NP-hardness of the no-wait problem $O2|\text{no-wait}|C_{\max}$ in which the processing of the second operation of each job must start immediately after the completion of its first operation.

Now let us turn to the cases of $Om||C_{\max}$ with $m \geq 3$ machines. As usual, the complexity jumps from easy to hard when we move from parameter 2 to parameter 3:

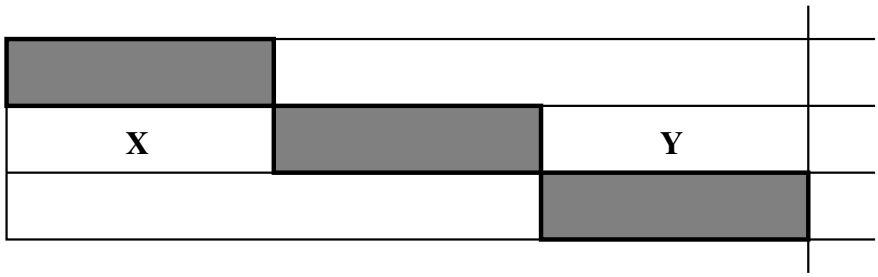

Figure 1 Illustration of the NP-hardness argument in Theorem 2.

▶ **Theorem 2** (Gonzalez & Sahni [10]). *For every fixed $m \geq 3$, problem $Om||C_{\max}$ is NP-hard.*

Proof. We only show hardness for $m = 3$. The proof is a polynomial time reduction from the NP-hard PARTITION problem [9]: *"Given k positive integers $q_1, \ldots, q_k$ with $\sum_{i=1}^{k} q_i = 2Q$, does there exist an index set $I \subseteq \{1, \ldots, k\}$ with $\sum_{i \in I} q_i = Q$?"*

For $j = 1, \ldots, k$ we create a job J_j with $p_{1,j} = p_{2,j} = p_{3,j} = q_j$. Furthermore, there is a job J_{k+1} with $p_{1,k+1} = p_{2,k+1} = p_{3,k+1} = Q$. We claim that the PARTITION instance has answer YES, if and only if the constructed instance of $O3||C_{\max}$ has a schedule with makespan at most $3Q$. The (only if part) is straightforward. For the (if part), consider the three operations of job J_{k+1} in a schedule with makespan $3Q$. By symmetry, we may assume that J_{k+1} is first processed on machine M_1, then on M_2, and finally on M_3. Then the second operation generates two time intervals X and Y of length Q on machine M_2; see Figure 1 for an illustration. The operations $O_{2,j}$ of the other jobs must be packed into intervals X and Y, and thereby yield a solution for the PARTITION instance. ◀

As the PARTITION problem is NP-hard in the weak sense, the argument in Theorem 2 only yields NP-hardness in the weak sense for $Om||C_{\max}$. The precise complexity (strong NP-hardness versus pseudo-polynomial time solvability) of problem $Om||C_{\max}$ is unknown. This complexity question has been open since the 1970s, and it forms the biggest and most important gap in our understanding of open shop scheduling.

▶ **Open problem 3.** *Prove that for every fixed number $m \geq 3$ of machines, problem $Om||C_{\max}$ is solvable in pseudo-polynomial time.*

Finally, let us discuss the complexity of problem $O||C_{\max}$ where the number of machines is specified as part of the input. An unpublished result of Lenstra [13] from the 1970s establishes strong NP-hardness of $O||C_{\max}$. Strong NP-hardness of $O||C_{\max}$ can also easily be deduced from a published result by Williamson & al. [24], who prove that $O||C_{\max}$ is NP-hard, even if all operations are of length $0, 1, 2$ and if the question is to decide whether there is a schedule with makespan 4.

4 A theorem on vector rearrangements

This section introduces an auxiliary problem and an auxiliary result that will be pivotal for the next section. Let $B \subset \mathbb{R}^d$ be the unit ball of a norm $\|\cdot\|$ on $\mathbb{R}^d$, that is, a d-dimensional closed convex body that is centrally symmetric about the origin. Suppose we are given n vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ that satisfy

$$\sum_{i=1}^{n} v_i = 0 \qquad \text{and} \qquad \|v_i\| \leq 1 \text{ for } 1 \leq i \leq n. \tag{2}$$

The goal is to find a permutation $v_{\pi(1)}, \ldots, v_{\pi(n)}$ of these vectors, so that for every k with $1 \leq k \leq n$ the norm $\|v_{\pi(1)} + v_{\pi(2)} + \cdots + v_{\pi(k)}\|$ of the partial sum is small. Steinitz [22] proved in 1913 that the norms of these partial sums can be bounded by a constant that only depends on the unit ball B (and Steinitz showed that the concrete constant $2d$ always works). The smallest such constant is called the *Steinitz constant* $c(B)$ of the norm.

▶ **Theorem 4** (Grinberg & Sevastianov [11]). *For any norm with unit ball $B \subset \mathbb{R}^d$, the Steinitz constant satisfies $c(B) \leq d$.*

The proof of Theorem 4 in [11] is an optimized and streamlined version of an earlier proof by Sevastianov [15]. It is extremely elegant, and we will sketch it now. Hence, let us consider vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ that satisfy (2). In a first step, we prove that there exists a system of subsets $V_d, V_{d+1}, \ldots, V_n$ of $\{v_1, \ldots, v_n\}$ that satisfies the following properties.

- $V_d \subseteq V_{d+1} \subseteq \cdots \subseteq V_n = \{v_1, \ldots, v_n\}$
- $|V_k| = k$ for $1 \leq k \leq n$
- There exist real numbers $\lambda_k(v) \in [0, 1]$ for $d \leq k \leq n$ and $v \in V_k$ with
  (A) $\sum_{v \in V_k} \lambda_k(v) = k - d$ for $d \leq k \leq n$, and
  (B) $\sum_{v \in V_k} \lambda_k(v) \cdot v = 0$ for $d \leq k \leq n$.

In other words, the coefficients $\lambda_k(v)$ constitute a linear dependency on V_k where all coefficients add up to $k - d$. The subsets V_k and the real numbers $\lambda_k(v)$ are constructed by a backward induction. For $k = n$, we have $V_n = \{v_1, \ldots, v_n\}$ and we define $\lambda_n(v) \equiv (n - d)/n$ for all v. These values satisfy condition (A) by definition, while condition (B) follows from (2).

Now assume that the sets $V_{k+1}, \ldots, V_n$ have already been defined together with the corresponding coefficients $\lambda_{k+1}(v), \ldots, \lambda_n(v)$. We consider the following system of linear constraints on $k + 1$ real variables $x(v)$ for $v \in V_{k+1}$.

$$\sum_{v \in V_{k+1}} x(v) \;=\; k - d \tag{3}$$

$$\sum_{v \in V_{k+1}} x(v) \cdot v \;=\; 0 \tag{4}$$

$$0 \leq x(v) \leq 1 \qquad \text{for } v \in V_{k+1} \tag{5}$$

Note that the system (3)–(5) is solvable, as can be seen for instance by setting

$$x(v) \;=\; \frac{k - d}{k + 1 - d}\, \lambda_{k+1}(v) \qquad \text{for } v \in V_{k+1}.$$

Hence the underlying $(k + 1)$-dimensional polytope is non-empty. Any extreme point x^* of this polytope must satisfy $k + 1$ of the linear constraints with equality. As constraint (3) yields one such equality and as constraint (4) yields d such equalities (one per dimension), in an extreme point at least $k + 1 - (d + 1) = k - d$ of the inequalities in (5) must be tight. Because of (3), this implies that in an extreme point x^* at least one of the variables $x^*(v)$ will be equal to zero. We construct the set V_k by dropping the corresponding vector v from V_{k+1} and by setting $\lambda_k(v) = x^*(v)$. This completes the construction of the subset system.

In the second step, we transform the subset system into the desired permutation. The first d vectors $v_{\pi(1)}, \ldots, v_{\pi(d)}$ are an arbitrary ordering of the vectors in set V_d. For $k \geq d+1$, we choose vector $v_{\pi(k)}$ as the unique element of $V_k - V_{k-1}$. We claim that in the resulting permutation, the norm of every partial sum $\sum_{i=1}^{k} v_{\pi(i)}$ is at most d. Indeed, for $k \leq d$ the triangle inequality together with $\|v_i\| \leq 1$ implies $\|\sum_{i=1}^{k} v_{\pi(i)}\| \leq \sum_{i=1}^{k} \|v_{\pi(i)}\| \leq d$. For

$d+1 \leq k \leq n$, the claim follows from the following chain of equations and inequalities, which is based on properties (A) and (B) and on the triangle inequality.

$$\begin{aligned} \|\sum_{i=1}^{k} v_{\pi(i)}\| &= \|\sum_{v \in V_k} v\| = \|\sum_{v \in V_k} v - \sum_{v \in V_k} \lambda_k(v) \cdot v\| \\ &\leq \sum_{v \in V_k} (1-\lambda_k(v)) \cdot \|v\| \leq \sum_{v \in V_k} (1-\lambda_k(v)) \\ &= |V_k| - \sum_{v \in V_k} \lambda_k(v) = k-(k-d) = d \end{aligned}$$

This completes the proof of Theorem 4. Note that the constructed permutation does not depend on the underlying norm. Note furthermore that the entire construction can easily be implemented in polynomial time.

Banaszczyk [2] slightly strengthened the bound in Theorem 4 on the Steinitz constants for norms in d-dimensional space to $c(B) \leq d-1+1/d$. Bergström [5] showed that the Steinitz constant of the Euclidean plane (2-dimensional space with Euclidean norm) equals $\sqrt{5}/2 \approx 1.118$. It is known (and easy to see) that the Steinitz constant of the d-dimensional Euclidean space is at least $\sqrt{d+3}/2$, and this might well be the correct value of the d-dimensional Euclidean Steinitz constant. However, at the current moment not even a sub-linear upper bound is known and the problem is wide open (even for $d=3$).

5 A tractable special case

Recall that $o_{\max}$ denotes the length of the longest operation, that $p_{\max}$ denotes the length of the longest job, and that $L_{\max}$ denotes the maximum machine load. Throughout this section we will assume that

$$L_1 = L_2 = L_3 = \cdots = L_m = L_{\max} \qquad \text{and} \qquad o_{\max} = 1. \tag{6}$$

The equality of all machine loads in (6) can be reached by adding dummy jobs, and $o_{\max}=1$ can be reached by scaling. We will apply the machinery for vector rearrangements (as described in the preceding section) to the open shop scheduling problem $Om||C_{\max}$. We introduce a unit ball B^* for a norm $\|\cdot\|_*$ in $(m-1)$-dimensional space, defined by

$$B^* = \left\{(x_1,\ldots,x_{m-1}) \in \mathbb{R}^{m-1} : \ |x_k| \leq 1 \text{ and } |x_k - x_\ell| \leq 1 \text{ for all } k \text{ and } \ell\right\}. \tag{7}$$

Every job J_j with processing times $p_{i,j}$ is translated into an $(m-1)$-dimensional vector

$$v_j = (p_{1,j} - p_{m,j},\ p_{2,j} - p_{m,j},\ \ldots,\ p_{m-1,j} - p_{m,j}). \tag{8}$$

Because of (6) we have $\sum_{j=1}^{n} v_j = 0$ and $\|v_j\|_* \leq 1$ for $1 \leq j \leq n$, so that the conditions (2) for the vector rearrangement Theorem 4 are satisfied. Consequently there exists a permutation $v_{\pi(1)},\ldots,v_{\pi(n)}$ of these vectors, so that

$$\|v_{\pi(1)} + v_{\pi(2)} + \cdots + v_{\pi(k)}\|_* \leq m-1 \qquad \text{for } k=1,\ldots,n. \tag{9}$$

We construct an infeasible schedule that on each machine processes the jobs without idle time in the ordering $J_{\pi(1)},\ldots,J_{\pi(n)}$; see Figure 2 for an illustration. This schedule is extremely infeasible, as it schedules all operations of every job into a short time interval; this is a consequence of (9) and the definition of norm $\|\cdot\|_*$. The positive effect of this type of

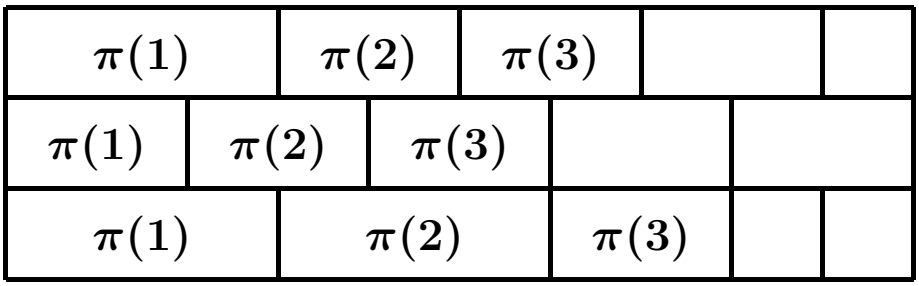

Figure 2 The infeasible schedule that results from the vector rearrangement.

infeasibility is that we have a good understanding of the global structure of this schedule. The completion time of operation $O_{i,j}$ in the infeasible schedule is denoted by $C_{i,j}$. Then for $k \geq 2$ we have

$$\begin{aligned} C_{1,\pi(k)} - C_{2,\pi(k-1)} &= \sum_{j=1}^{k} p_{1,\pi(j)} - \sum_{j=1}^{k-1} p_{2,\pi(j)} \\ &= \sum_{j=1}^{k-1} (p_{1,\pi(j)} - p_{2,\pi(j)}) + p_{1,\pi(k)} \;\leq\; (d-1)+1 \;=\; d. \end{aligned}$$

In the inequality, we use (9) and $p_{1,\pi(k)} \leq o_{\max} = 1$ from (6). By applying analogous arguments, we derive

$$\begin{aligned} \Delta_1 &:= \max_{k\geq 2} C_{m,\pi(k)} - C_{1,\pi(k-1)} &\leq\; m \\ \Delta_2 &:= \max_{k\geq 2} C_{1,\pi(k)} - C_{2,\pi(k-1)} &\leq\; m \\ \Delta_3 &:= \max_{k\geq 2} C_{2,\pi(k)} - C_{3,\pi(k-1)} &\leq\; m \\ &\quad \cdots \quad \cdots \quad \cdots \\ \Delta_m &:= \max_{k\geq 2} C_{m-1,\pi(k)} - C_{m,\pi(k-1)} &\leq\; m \end{aligned}$$

This means that we can turn the infeasible schedule into a feasible schedule, by simply shifting all operations on every machine M_i by $(i-1)m$ time units into the future. The makespan of the resulting schedule will be bounded by $L_{\max} + (m-1)o_{\max}$, which yields a reasonable approximation result. We will describe next how to get an even better result. We wrap the infeasible schedule around a cylinder with circumference $L_{\max}$. Each of the individual machine schedules forms a ring around the cylinder that may be rotated. We freeze the ring for machine M_1, and we mark the starting time of job $J_{\pi(1)}$ as the zero-point.

We rotate the ring for machine M_2 by Δ_2 time units and thereby shift the starting time of each operation by Δ_2 into the future. By doing this, we resolve all collisions between operations on M_1 and operations on M_2: Every job has now disjoint processing intervals on M_1 and M_2. Then we rotate the ring for machine M_2 by $\varepsilon_2 \leq o_{\max}$ additional time units, so that one of the operations on M_2 is started at the marked zero-point. Next, we do a similar rotation of the ring for machine M_3 by $\Delta_2 + \Delta_3 + \varepsilon_2 + \varepsilon_3$ time units, so that all collisions between M_2 and M_3 are resolved and so that one of the operations on M_3 is started at the marked zero-point. And so on. The ring for machine M_i is rotated by $\sum_{k=2}^{i} \Delta_k + \varepsilon_k$ time units, so that all collisions are resolved and so that some operation starts at the zero-point.

In the end, we cut the rings open at the marked zero-point and flatten them into a schedule for the considered open shop instance. If $L_{\max} - \Delta_1$ is larger than the total length of all shifts, the resulting schedule will be feasible: Before the shifting, all operations of job J_j were scheduled very close to each other in time. The first shift puts $O_{1,j}$ and $O_{2,j}$ into disjoint processing intervals, and each of the later shifts puts another operation into a disjoint processing interval. As $L_{\max}$ is sufficiently large, the later operations of job J_j will not be shifted all the way around the cylinder and hence cannot cause collisions with the first operation $O_{1,j}$ of that job. Since $\Delta_i \leq m$ and since $\varepsilon_i \leq o_{\max} \leq 1$, the total length of all shifts is at most $(m-1)(m+1)$, and this should be at most $L_{\max} - \Delta_1 \geq L_{\max} - m$.

▶ **Theorem 5** (Fiala [8]). *If an instance of $Om||C_{\max}$ satisfies $L_{\max} \geq (m^2+m-1)o_{\max}$, then the optimal makespan is $L_{\max}$. Furthermore, an optimal schedule can be computed in polynomial time.*

One consequence of Theorem 5 is that open shop problems in the real world are often easy to solve: If all jobs are drawn from the same distribution and if there is a sufficiently large number of jobs, then the condition $L_{\max} \geq (m^2+m-1)o_{\max}$ in Theorem 5 will hold true and the instances become easy to solve.

Belov & Stolin [4] were the first to apply vector rearrangement methods in the area of scheduling (and they applied them to the flow shop problem). Fiala [8] discovered the nice connection to the open shop problem; he actually proved a much stronger version of Theorem 5 where the factor m^2+m-1 is replaced by $8m' \log_2(m') + 5m'$ where m' is the smallest power of 2 greater or equal to m. Bárány & Fiala [3] improved Fiala's bound by a factor of 2, and Sevastianov [16] improved it down to roughly $(16/3)m \log_2 m$. Sevastianov [17] surveys and summarizes the history of vector rearrangement methods in the area of scheduling.

In the light of the above results, it is natural to ask for the smallest value $\eta(m)$, so that every instance of $Om||C_{\max}$ with $L_{\max} \geq \eta(m)o_{\max}$ automatically satisfies $C^*_{\max} = L_{\max}$.

▶ **Open problem 6.** *Derive good upper and lower bounds on $\eta(m)$ for $m \geq 3$.*

Sevastianov [18] establishes the upper bound $\eta(m) \leq m^2 - 1 + 1/(m-1)$, which is useful for small values of m. [18] also establishes the lower bound $\eta(m) \geq 2m-2$. Here is the bad instance for $m=3$ machines that demonstrates $\eta(3) \geq 4$: There is one job with processing time 1 on each machine. Furthermore, for each machine M_i $(i = 1,2,3)$ there are three jobs with processing time $1-\varepsilon$ on M_i and processing time 0 on the other two machines. Then $L_{\max} = 4 - 3\varepsilon$ and $C^*_{\max} = 4 - \varepsilon$.

Sevastianov [18] also shows that $Om||C_{\max}$ remains NP-hard, if it is restricted to instances with $L_{\max}/o_{\max} = \rho$ where $1 < \rho < 2m-3$. It is not clear, what is going on for instances with $2m-3 \leq \rho < \eta(m)$. Perhaps, the instances with $\rho < \eta(m)$ are all NP-hard; in that case $\eta(m)$ would be a threshold at which the complexity jumps from NP-hard to trivial.

▶ **Open problem 7.** *Determine the computational complexity of the restriction of $Om||C_{\max}$ to instances with $L_{\max}/o_{\max} = 2m-2$.*

6 Approximation for an arbitrary number of machines

Here is a simple greedy algorithm for $O||C_{\max}$: Start at time $t=0$, and whenever some machine becomes idle and there is some job available that still needs processing on that machine then assign that job to that machine. Ties are broken arbitrarily. This greedy algorithm was formulated by Bárány & Fiala [3] who attribute it to private communication with Anna Racsmány.

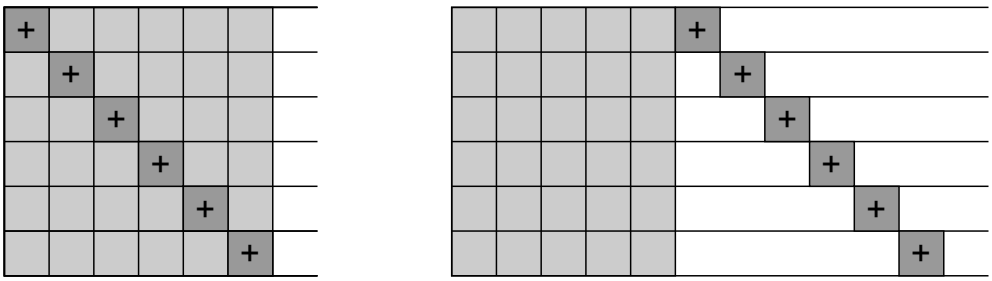

Figure 3 A lower bound instance for the greedy algorithm on $m = 6$ machines. The dummy jobs are shown in light-gray, while the operations of job J^+ are in dark-gray and marked by $+$.

▶ **Theorem 8** (Bárány & Fiala [3]). *The greedy algorithm is a polynomial time approximation algorithm for $O||C_{\max}$ with worst case ratio at most 2.*

Proof. Consider a greedy schedule, and let $O_{i,j}$ be an operation that completes last. Then on machine M_i, the greedy schedule has busy time intervals and idle time intervals. The total length of the busy time intervals is $L_i \leq L_{\max}$. Whenever M_i is idle, it is not processing job J_j and the only possible reason for this is that job J_j is being processed on one of the other machines. Therefore the total length of the idle time intervals is at most $p_j \leq p_{\max}$. This implies that the greedy makespan is at most $L_{\max} + p_{\max}$, which according to (1) is bounded by $2\beta^* \leq 2C^*_{\max}$. ◀

The result in Theorem 8 can also be derived as a corollary to a more general result by Aksjonov [1]. How good is the worst case analysis in this theorem? Consider the following instance with m machines and $m^2 - m + 1$ jobs. For $i = 1, \ldots, m$ there are $m - 1$ dummy jobs that each need one unit of processing time on machine M_i and zero processing time on all other machines. Furthermore, there is a job J^+ that needs one unit of processing time on every machine. There is a greedy schedule with makespan $2m - 1$ for this instance, in which from time 0 to time $m - 1$ all machines are busy with processing the dummy jobs, and from time $m - 1$ to time $2m - 1$ they process job J^+. As the optimal makespan is $C^*_{\max} = m$, the worst case ratio of the greedy algorithm is at least $2 - 1/m$; see Figure 3 for an illustration. Chen & Strusevich [6] have settled the cases $m = 2$ and $m = 3$ of the following open problem by a tedious case analysis, and Chen & Yu [7] have settled the case $m = 4$.

▶ **Open problem 9.** *Prove that the greedy algorithm for $Om||C_{\max}$ has worst case ratio at most $2 - 1/m$.*

A difficult open problem in this area asks whether there is a polynomial time approximation algorithm for $O||C_{\max}$ with worst case ratio strictly better than 2. One possible approach would work with the lower bound β^* defined in (1). Sevastianov & Tchernykh [19] have proved $C^*_{\max} \leq 4\beta^*/3$ for problem $O3||C_{\max}$. Their proof is based on heavy case analysis and on case enumeration with the help of a computer program. As the computer program described in [19] takes more than 200 hours of computation time, this approach does not seem to be applicable to $m \geq 4$ machines.

▶ **Open problem 10.** *Prove that any instance of $Om||C_{\max}$ satisfies $C^*_{\max} \leq 3\beta^*/2$.*

Here is an instance that demonstrates that the factor 3/2 in this open problem would in fact be best possible. The instance uses m machines and $m + 1$ jobs. For $j = 1, \ldots, m$ the job J_j consists of the operation O_{jj} with processing time $p_{jj} = m - 1$ on machine M_j, and with

processing times 0 on the other $m-1$ machines. The final job J_{m+1} has m operations all with processing time 1. Then $\beta^* = m$ and $C^*_{\max} = \lceil m/2 \rceil + m - 1$. As m becomes large, the ratio between $C^*_{\max}$ and β^* tends to $3/2$.

Now let us turn to negative results on the worst case ratio of polynomial time approximation algorithms for $O||C_{\max}$. Williamson & al. [24] prove that it is NP-hard to decide whether an $O||C_{\max}$ instance with integer processing times has optimal makespan at most 4. Since the optimal makespan of a NO-instance is at least 5, a polynomial time approximation algorithm with worst case ratio $5/4 - \varepsilon$ would allow us to distinguish the YES-instances from the NO-instances in polynomial time.

▶ **Theorem 11** (Williamson & al. [24]). *Unless P=NP, problem $O||C_{\max}$ does not allow a polynomial time approximation algorithm with worst case ratio strictly better than $5/4$.*

It might be possible to lift the hardness proof in [24] to get stronger inapproximability results.

▶ **Open problem 12.** *Analyze the computational complexity of the (a, b)-versions of $O||C_{\max}$ instances with integer processing times: Decide whether the optimal makespan does satisfy $C^*_{\max} \leq a$ or whether it does satisfy $C^*_{\max} \geq b$.*

If this (a, b)-version turns out to be NP-hard for fixed integers a and b, then $O||C_{\max}$ cannot have a polynomial time approximation algorithm with worst case ratio strictly better than b/a unless $P = NP$. The result in [24] yields NP-hardness of the $(4, 5)$-version, and [24] also shows that the $(3, 4)$-version is solvable in polynomial time. Hence the smallest interesting open cases would concern the $(5, 7)$-version and the $(6, 8)$-version.

7 Approximation for a fixed number of machines

For an arbitrary number of machines, polynomial time approximation algorithms cannot have worst case ratios very close to 1; see Theorem 11. For a fixed number of machines, the situation is much better and there is a polynomial time approximation scheme (PTAS).

▶ **Theorem 13** (Sevastianov & Woeginger [20]). *For every fixed $m \geq 3$, problem $Om||C_{\max}$ has a PTAS.*

We now show a proof of Theorem 13 for the special case $m = 3$. (The general case is based on exactly the same ideas, while some of the details become a bit messier.) So let us consider some instance of $O3||C_{\max}$, and let ε with $0 < \varepsilon < 1$ be some small real number that indicates the desired precision of approximation. The running time of our algorithm will be polynomial in the instance size, but exponential in $1/\varepsilon$. The resulting makespan will come arbitrarily close to $C^*_{\max}$, if we let ε tend to 0.

As often in approximation schemes for scheduling problems, the jobs are classified into *big* and into *small* jobs. We call a job *big*, if one of its operations has length $p_{i,j} \geq \varepsilon\beta^*$, where β^* is the lower bound defined in (1). All other jobs are *small* jobs, and we want to assume for the moment that (***) all operations of all small jobs have length $p_{i,j} \leq \varepsilon^2\beta^*$; we will show later how to work around this assumption. Since the total processing time of all jobs is at most $3L_{\max} \leq 3\beta^*$ and as every big job has processing time at least $\varepsilon\beta^*$, there are at most $3/\varepsilon$ big jobs. The algorithm now works in two phases.

- In the first phase, we determine an optimal schedule for the big jobs. This can be done in $O(1)$ time, as the running time does only depend on $1/\varepsilon$ and does not depend on the instance size. In the resulting schedule, every machine processes at most $3/\varepsilon$ operations with at most $3/\varepsilon$ gaps of idle time between the operations; see Figure 4 for an illustration.

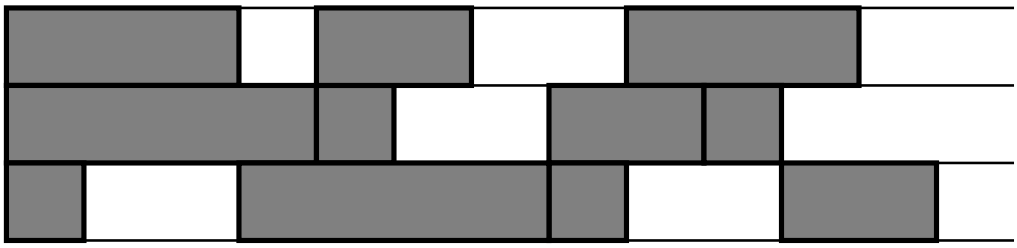

Figure 4 An optimal schedule for the big jobs in the proof of Theorem 13.

- In the second phase, we pack the operations of the small jobs into the idle gaps. This is done greedily (as in Theorem 8). Start at time $t = 0$, and whenever some machine becomes idle at some time t, try to process an unprocessed small operation on that machine. There are two possible scenarios under which an unprocessed small operation $O_{i,j}$ cannot be started at time t: First another operation O_{kj} of the same job might currently be processed on some other machine. Secondly, the remaining part of the current gap might be too small to accommodate $O_{i,j}$. If one of these scenarios occurs, we try to schedule some other small operation. If no operation can be scheduled, then the machine is left idle for the moment.

Now let us analyze the makespan $C^A_{\max}$ of the resulting approximating schedule. Let $O_{i,j}$ be an operation that completes last. In the first case assume that $O_{i,j}$ belongs to a big job. Then $C^A_{\max}$ coincides with the optimal makespan for the big jobs, and we actually have found an optimal schedule. In the second case assume that $O_{i,j}$ belongs to a small job. Then we consider the busy time intervals and the idle time intervals on machine M_i. The total length of all busy time intervals is the load $L_i \leq \beta^*$. Whenever machine M_i was idle, it could not process operation $O_{i,j}$. This means that either (i) job J_j was being processed on one of the other machines, or that (ii) the remaining gap was too small to accommodate $O_{i,j}$. The total idle time of type (i) is bounded by the length of the small job J_j, which is at most $3\varepsilon^2\beta^*$. The total idle time of type (ii) is bounded by the number of gaps multiplied by the length of operation $O_{i,j}$, which is at most $(3/\varepsilon) \cdot (\varepsilon^2\beta^*) = 3\varepsilon\beta^*$. Altogether, this implies that the approximating makespan can be bounded as

$$C^A_{\max} \;=\; \text{Busy} + \text{Idle(i)} + \text{Idle(ii)} \;\leq\; \beta^* + 3\varepsilon^2\beta^* + 3\varepsilon\beta^* \;\leq\; (1 + 3\varepsilon^2 + 3\varepsilon)\, C^*_{\max}. \qquad (10)$$

As ε tends to 0, the error factor $1+3\varepsilon^2+3\varepsilon$ tends to 1. This yields the desired PTAS modulo the assumption (***).

It remains to discuss what to do with assumption (***), which essentially postulates an empty no man's land between big operations (of length at least $\varepsilon\beta^*$) and small operations (of length at most $\varepsilon^2\beta^*$). In other words, under assumption (***) non-big jobs must not contain operations of intermediate length $\varepsilon^2\beta^* < p_{i,j} < \varepsilon\beta^*$. This assumption will be totally wrong for most instances, but we can come very close to it by playing around with the value of ε. This is done as follows.

For a real number δ with $0 < \delta < 1$, we say that a job is *δ-big*, if one of its operations has length $p_{i,j} \geq \delta\beta^*$ and otherwise it is *δ-small*. An operation $O_{i,j}$ is *δ-nasty*, if it belongs to a δ-small job and satisfies the inequality $\delta^2\beta^* < p_{i,j} < \delta\beta^*$. By $N(\delta)$ we denote the total length of all δ-nasty operations. Now consider the real numbers $\delta_k = \varepsilon^{2^k}$ for $k \geq 0$. Then every operation $O_{i,j}$ is δ_k-nasty for at most one choice of index k. We search for an index k that satisfies the inequality

$$N(\delta_k) \;\leq\; \varepsilon\beta^*. \qquad (11)$$

If some δ_k violates (11), then the corresponding δ_k-nasty operations consume at least $\varepsilon\beta^*$ of the total processing time of all operations (which is at most $3\beta^*$). Hence some $k \leq 3/\varepsilon$ will indeed satisfy (11). From now on we work with that particular index k and with that particular value δ_k.

The final approximation scheme works as follows. First we remove from the instance all the δ_k-small jobs that contain some δ_k-nasty operation. To the surviving jobs we apply the original approximation algorithm as described above with $\varepsilon := \delta_k$, and thereby find a schedule with makespan at most $(1 + 3\delta_k^2 + 3\delta_k)\, C^*_{\max}$ according to (10). In the end, we greedily add the previously removed jobs with δ_k-nasty operations to this schedule. Since the overall processing time of all removed jobs is at most $3\varepsilon\beta^*$, this increases the makespan by at most $3\varepsilon\beta^*$. Since $\delta_k \leq \varepsilon$, this altogether yields a schedule of makespan at most $(1 + 3\varepsilon^2 + 6\varepsilon)C^*_{\max}$. This completes the proof of Theorem 13 for the special case $m = 3$.

An FPTAS (fully polynomial time approximation scheme) is a PTAS whose time complexity is also polynomially bounded in $1/\varepsilon$. The following open problem is closely linked to the existence of pseudo-polynomial time exact algorithms for $Om||C_{\max}$.

▶ **Open problem 14.** *Prove that problem $Om||C_{\max}$ has an FPTAS for every fixed $m \geq 3$.*

References

1. V.A. Aksjonov. A polynomial-time algorithm for an approximate solution of a scheduling problem (in Russian). *Upravlyaemye Sistemy*, 28:8–11, 1988.
2. W. Banaszczyk. The Steinitz constant of the plane. *Journal für die Reine und Angewandte Mathematik*, 373:218–220, 1987.
3. I. Bárány and T. Fiala. Nearly optimum solution of multimachine scheduling problems (in Hungarian). *Szigma Mathematika Közgazdasági Folyóirat*, 15:177–191, 1982.
4. I.S. Belov and Ya. N. Stolin. An algorithm for the single-route scheduling problem (in Russian). In *Matematiceskaja ékonomika i funkcionalnyi analiz (Mathematical Economics and Functional Analysis)*, pages 248–257. Nauka, Moscow, 1974.
5. V. Bergström. Zwei Sätze über ebene Vektorpolygone. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 8:148–152, 1931.
6. B. Chen and V.A. Strusevich. Approximation algorithms for three-machine open shop scheduling. *ORSA Journal on Computing*, 5:321–326, 1993.
7. B. Chen and W. Yu. How good is a dense shop schedule? *Acta Mathematicae Applicatae Sinica*, 17:121–128, 2001.
8. T. Fiala. An algorithm for the open-shop problem. *Mathematics of Operations Research*, 8:100–109, 1983.
9. M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
10. T. Gonzalez and S. Sahni. Open shop scheduling to minimize finish time. *Journal of the Association for Computing Machinery*, 25:92–101, 1976.
11. V. Grinberg and S.V. Sevastianov. The value of the Steinitz constant (in Russian). *Funktionalnyi Analiz i ego Prilozheniia (Functional Analysis and Its Applications)*, 14:125–126, 1980.
12. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin, editors, *Logistics of Production and Inventory (Handbooks in Operations Research and Management Science)*, pages 445–522. North-Holland, Amsterdam, 1993.
13. J.K. Lenstra. Strong NP-hardness of open shop scheduling. Unpublished manuscript, 1978.
14. S. Sahni and Y. Cho. Complexity of scheduling shops with no wait in process. *Mathematics of Operations Researc*, 4:448–457, 1979.

15 S.V. Sevastianov. Approximate solution of some problems in scheduling theory (in Russian). *Metody Diskretnogo Analiza*, 32:66–75, 1978.

16 S.V. Sevastianov. A polynomially solvable case of the open shop problem with arbitrary number of machines (in Russian). *Kibernetika i Systemnii Analiz*, 6:135–154, 1992.

17 S.V. Sevastianov. On some geometric methods in scheduling theory: a survey. *Discrete Applied Mathematics*, 55:59–82, 1994.

18 S.V. Sevastianov. Vector summation in Banach space and polynomial algorithms for flow shops and open shops. *Mathematics of Operations Research*, 20:90–103, 1995.

19 S.V. Sevastianov and I.D. Tchernykh. Computer-aided way to prove theorems in scheduling. In *Proceedings of the 6th European Symposium on Algorithms (ESA'1998)*, LNCS 1461, pages 502–513. Springer, 1998.

20 S.V. Sevastianov and G.J. Woeginger. Makespan minimization in open shops: A polynomial time approximation scheme. *Mathematical Programming*, 82:191–198, 1998.

21 N.V. Shaklevich and V.A. Strusevich. Two machine open shop scheduling problem to minimize an arbitrary machine usage regular penalty function. *European Journal of Operational Research*, 70:391–404, 1993.

22 E. Steinitz. Bedingt konvergente Reihen und konvexe Systeme. *Journal für die Reine und Angewandte Mathematik*, 143:128–176, 1913.

23 M. van den Akker, J.A. Hoogeveen, and G.J. Woeginger. The two-machine open shop problem: to fit or not to fit, that is the question. *Operations Research Letters*, 31:219–224, 2003.

24 D.P. Williamson, L.A. Hall, J.A. Hoogeveen, C.A.J. Hurkens, J.K. Lenstra, S.V. Sevastianov, and D.B. Shmoys. Short shop schedules. *Operations Research*, 45:288–294, 1997.