The two-machine open shop problem: to fit, or not to fit, that was the question *

Marjan van den Akker
Department of Computer Science
Utrecht University
The Netherlands

Han Hoogeveen
Department of Computer Science
Utrecht University
The Netherlands

Gerhard J. Woeginger
Institut für Mathematik B
TU Graz
Austria
Department of Mathematics
University of Twente
The Netherlands

May 14, 2002

Abstract

We consider an open shop scheduling problem with two machines. Here each job consists of two operations, and it is prescribed that the first (second) operation has to be excuted by the first (second) machine. The order in which the two operations have to be scheduled is not fixed, but their execution intervals cannot overlap. The question that we are interested in is whether, given values D_1 and D_2, there exists a feasible schedule such that the first and second machine can process all jobs in the intervals $[0, D_1]$ and $[0, D_2]$, respectively. We present four necessary conditions for D_1 and D_2, which can be computed in $O(n)$ time, and show that these conditions are sufficient as well by specifying an algorithm that constructs a feasible schedule if D_1 and D_2 satisfy these conditions. We further show that there are at most two nondominated points (D_1, D_2) for which there exists a schedule that fits.

Keywords. Scheduling, sequencing, open shop scheduling, flow shop scheduling, bicriterion optimization.

1 Introduction

We consider the following machine scheduling problem. There are two machines M_1 and M_2 that can process at most one job at a time. The machines are continuously available from time zero to time D_1 (M_1) and D_2 (M_2), where D_1 and D_2 are two given integers. These machines have to process a given job set $\mathcal{J}$ that consists of n jobs $J_1, \ldots, J_n$. Each job J_j consists of two

*This work was partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

operations, where the first (second) one has to be processed by M_1 (M_2) for an uninterrupted period of length a_j (b_j). There is no order prescribed in which the two operations should be executed, but the operations of one job are not allowed to overlap in their execution. In the literature such a scheduling environment is known as a two-machine *open shop*, and it is encoded by the entry $O2$ in the three-field notation scheme of Graham, Lawler, Lenstra, and Rinnooy Kan (1979). The open shop can be viewed upon as a relaxation of the *flow shop* environment, in which each job first has to visit M_1 and then M_2; this environment is encoded by the entry $F2$. For both the open shop and flow shop problem, there is only one optimization problem that can be solved to optimality in polynomial time: minimizing the makespan, which is defined as the time at which the last job is completed. These problems are encoded as $O2||C_{\max}$ and $F2||C_{\max}$. The first one is solvable through the algorithm by Gonzalez and Sahni (1976), which runs in $O(n)$ time. The second problem is solved through the famous algorithm developed by Johnson (1954), which runs in $O(n \log n)$ time. As we will heavily use Gonzalez and Sahni's algorithm, we will state this algorithm in Section 2. In our case, we do not have an optimization problem, but a feasibility problem, although we can easily turn it into an optimization problem by minimizing some objective function $f(D_1, D_2)$, if we do no longer consider the values D_1 and D_2 as given. The makespan criterion can then be modeled through $f(D_1, D_2) = \max\{D_1, D_2\}$.

This paper is organized as follows. In Section 2, we repeat the algorithm by Gonzalez and Sahni, which is a prerequisite for Section 3, in which we show that it can be checked in $O(n)$ time, whether there exists a feasible schedule for our problem, by establishing four necessary and sufficient conditions for D_1 and D_2. Finally in Section 4 we show that for each instance there exist only two nondominated or Pareto optimal points (D_1, D_2) for which there exists a feasible schedule meeting D_1 and D_2, and we will show how these can be determined in $O(n)$ time.

2 Gonzalez and Sahni's algorithm

In this section we describe Gonzalez and Sahni's algorithm that solves $O2||C_{\max}$. Gonzalez and Sahni start with the following three lower bounds on the minimum makespan:

- the total processing time $\sum a_j$ of all jobs on M_1;
- the total processing time $\sum b_j$ of all jobs on M_2; and
- the maximum total processing time $\max(a_j + b_j)$ per job.

Subsequently, they present an $O(n)$ algorithm that finds a feasible schedule with a makespan that is equal to the maximum of the three lower bounds; hence, this makespan is mimimal. Their algorithm is as follows.

Gonzalez and Sahni's Algorithm

Step 0. Set $q \leftarrow \arg\max(a_j + b_j)$. If $a_q + b_q \geq \max\{\sum a_j, \sum b_j\}$, then schedule J_q on M_1 (M_2) in the interval $[0, a_q]$ ($[a_q, a_q + b_q]$) and schedule the remaining jobs in $[a_q, \sum a_j]$ on M_1 and in $[0, \sum b_j - b_q]$ on M_2. Stop.

Step 1. Define $A = \{J_j | a_j \geq b_j\}$ and $B = \{J_j | a_j < b_j\}$. Choose any two distinct jobs J_l and J_r such that

$$a_r \geq \max_{J_j \in A} b_j \quad \text{and} \quad b_l \geq \max_{J_j \in B} a_j.$$

Let $A' = A\backslash\{J_l, J_r\}$ and $B' = B\backslash\{J_l, J_r\}$.

Step 2. If $\sum a_j - a_l \geq \sum b_j - b_r$, then construct a feasible schedule where:

- M_1 first executes J_l, then all jobs from B' in any order, all jobs from A' in any order, and finally J_r.
- M_2 executes the jobs in the order J_r, J_l, all jobs from B' in any order, and finally all jobs from A' in random order. The execution intervals are as follows:
  - if $a_r \leq \sum b_j - b_r$, then both machines process the jobs contiguously (not necessarily starting at time zero), such that on both machines the last operation ends at time $\max\{\sum a_j, \sum b_j\}$;
  - if $a_r > \sum b_j - b_r$, then M_1 is continuously busy processing in the entire interval $[0, \sum a_j]$, whereas M_2 is idle until time $\sum a_j - a_r - b_r$ and then processes the jobs contiguously.

Step 3. If $\sum a_j - a_l < \sum b_j - b_r$, then construct a feasible schedule where:

- M_1 starts with all jobs from B' in any order, then all jobs from A' in any order, J_r, and finally J_l;
- M_2 first executes J_l, then all jobs from B' in any order, then the jobs from A' in any order, and finally J_r. The execution intervals are as follows:
  - if $a_l \leq \sum b_j - b_l$, then both machines process the jobs contiguously (not necessarily starting at time zero), such that on both machines the last operation ends at time $\max\{\sum a_j, \sum b_j\}$;
  - if $a_l > \sum b_j - b_l$, then M_1 is continuously busy processing in the entire interval $[0, \sum a_j]$, whereas M_2 is idle until time $\sum a_j - a_l - b_l$ and then processes the jobs contiguously.

Theorem 2.1 *Gonzalez and Sahni's algorithm finds a feasible schedule with makespan equal to the maximum of the three given lower bounds.* ■

3 Finding a schedule that fits

In this section we consider the problem of determining for two given values D_1 and D_2 whether there exists a feasible schedule such that M_1 (M_2) can process all tasks in the interval $[0, D_1]$ ($[0, D_2]$). We present necessary and sufficient conditions for D_1 and D_2. We present an algorithm that constructs in $O(n)$ time a feasible schedule that meets D_1 and D_2 given that D_1 and D_2 obey the conditions.

The case $D_1 = D_2$ boils down to the problem of minimizing the makespan, which is solved through Gonzalez and Sahni's algorithm presented in the previous section.

We proceed with the case that $D_1 \neq D_2$. Since the role of the machines is interchangeable (there is no prescribed ordering in the execution of the operations), we assume without loss of generality that $D_1 < D_2$. Gonzalez and Sahni's lower bounds immediately lead to three necessary conditions on D_1 and D_2. We find that D_1 and D_2 must satisfy

$$\sum_{j=1}^{n} a_j \leq D_1. \tag{1}$$

$$\sum_{j=1}^{n} b_j \leq D_2. \tag{2}$$

$$\max_{1 \leq j \leq n} (a_j + b_j) \leq D_2. \tag{3}$$

Unfortunately, Conditions 1–3 are not sufficient. Consider the following two-job example with processing times 1 on M_1 and 2 on M_2. The combination $D_1 = 2$ and $D_2 = 4$ satisfies all conditions, but there is no feasible schedule that fits. It is easily checked that, either $(D_1, D_2) \geq (3, 4)$, or $(D_1, D_2) \geq (2, 5)$ is required for a fitting feasible schedule. This example leads to the following lower bound on D_2 for a given value D_1. Define S as the subset of $\mathcal{J}$ that contains all jobs J_j with $a_j + b_j > D_1$. Since these jobs cannot be completed at time D_1, they must be executed by M_1 first and then by M_2. Ignoring the jobs that are not in S, we obtain a two-machine flow shop problem, which can be solved through Johnson's algorithm in $O(n \log n)$ time. Since the jobs in S constitute a special instance, we can solve it in $O(n)$ time, however. The specialty of the instance lies in the fact that we can complete each job on M_1 at time $\sum a_j \leq D_1$, whereas the first job in any feasible schedule is completed after time D_1 on M_2 by definition of S. Hence, we only care about selecting the right job to be processed first, after which we can process the remaining jobs in any order on M_1 and M_2 without any idle time. If we start with some job J_0, then we find a schedule with makespan equal to $a_0 + \sum_{j \in S} b_j$; hence, we must select the job J_j in S with minimum a_j to be processed first. This gives us Condition 4 on D_2:

$$D_2 \geq \min_{J_j \in S(D_1)} a_j + \sum_{J_j \in S(D_1)} b_j \equiv F^*(D_1); \tag{4}$$

we have added the argument $S(D_1)$ to show the dependence between S and D_1. In our example above, we have for $D_1 = 2$ that $S(2) = \{J_1, J_2\}$ and we find $F^*(2) = 5$. If $D_1 = 3$, then $S(3) = \emptyset$ and hence $F^*(3) = 0$.

In the remainder of this section we will show that the Conditions 1 – 4 are necessary (which should not come as a surprise) and sufficient. We present an algorithm that constructs a schedule in which M_1 and M_2 are finished at times D_1 and D_2 if the conditions are satisfied. The basis is formed by adapting the schedule determined by Gonzalez and Sahni's algorithm.

Suppose that we are given values D_1 and D_2 that satisfy the Conditions 1 – 4. Without loss of generality, we assume that we cannot decrease D_1 or D_2 without violating at least one of these conditions. We will design an algorithm that constructs a feasible schedule meeting D_1 and D_2. The algorithm breaks up the instance set in a set of categories, and for each category we describe how a feasible algorithm can be found.

We start with instances in which there exists a job J_q with $a_q + b_q \geq \max\{\sum a_j, \sum b_j\}$. This corresponds to Step 0 in Gonzalez and Sahni's algorithm, and the cure is the same: We process job J_q in $[0, a_q]$ on M_1 and in $[a_q, a_q + b_q]$ on M_2, after which we schedule the remaining jobs in the remaining gaps. The feasibility of this schedule follows immediately from Conditions 1 – 3.

We proceed as described in Gonzalez and Sahni's algorithm and find the jobs J_l and J_r. Suppose that $\sum a_j - a_l \geq \sum b_j - b_r$, which leads us to Step 2 of their algorithm. We increase a_r by $D_2 - D_1$ (we denote the adjusted a_j-values by $\bar{a}_j$) and construct the schedule as described by the algorithm. In both cases M_1 and M_2 finish processing at time $\max\{\sum \bar{a}_j, \sum b_j\} \leq D_2$ and J_r is processed last on M_1. If we replace the processing time $\bar{a}_r$ by a_r, then we have a feasible schedule meeting D_1 and D_2.

Now suppose that $\sum a_j - a_l > \sum b_j - b_r$, which leads us to Step 3 of Gonzalez and Sahni's algorithm. Since we will now end up with a schedule in which J_l is processed last by M_1, it looks promising to increase a_l by $D_2 - D_1$ (again, we denote the adjusted a_j-values by $\bar{a}_j$). Subsequently, we construct the schedule as described in Step 3. The case that $\bar{a}_l \leq \sum b_j - b_l$ is a piece of cake: both machines are finished at time D_2 and replacing the processing time $\bar{a}_l$ by a_l does the trick. In the case that $\bar{a}_l > \sum b_j - b_l$, we have that M_2 starts processing the jobs at time $\sum \bar{a}_j - \bar{a}_l - b_l$. In Gonzalez and Sahni's algorithm, we had already filtered out the case that $a_l + b_l \geq \max\{\sum a_j, \sum b_j\}$ and hence we could guarantee that M_2 did not start before time zero. If $\sum \bar{a}_j - \bar{a}_l - b_l \geq 0$, then we can apply our, by now standard, trick and end up with a feasible schedule that meets the bounds. If $\sum \bar{a}_j - \bar{a}_l - b_l < 0$, then we have to shift the whole schedule such that M_2 starts at time zero; if we then reduce the processing time of J_l on M_1, then we find that M_1 finishes processing at time $\bar{a}_l + b_l - \sum \bar{a}_j + \sum a_j = a_l + b_l$. Hence, if $a_l + b_l \leq D_1$, then we are done; otherwise, we cannot use Gonzalez and Sahni's algorithm anymore. Obviously, it is now time for Condition 4 to come into play.

Define S as the subset of $\mathcal{J}$ that contains all jobs J_j with $a_j + b_j > D_1$. As we have just observed, $J_l \in S$. Moreover, we know that when S contains no jobs with $a_j + b_j \leq D_1$, then solving $F2||C_{\max}$ for S yields a makespan that is smaller than or equal to D_2 (this is Condition 4). We construct our schedule through an iterative process, where we augment S in each iteration until we have a feasible schedule that fits. We use $a(S)$ and $b(S)$ as a short hand notation for the total processing time of the jobs in S on M_1 and M_2, respectively. We first check whether $a(S) + b(S) \geq D_2$. If this is the case, then we can find a feasible schedule by processing the jobs in S in random order but starting with the one with minimum a_j-value among the jobs with $a_j + b_j > D_1$ in the intervals $[0, a(S)]$ and $[D_2 - b(S), D_2]$ on M_1 and M_2, after which we put the remaining jobs in the gaps. If this is not the case, then we check whether there exists a job J_v with $J_j \notin S$ such that $b_v \geq a(S)$. If J_v exists, then we construct a feasible schedule in the following way:

- M_1 starts at time zero with the jobs in S followed by all jobs not in S except for J_v; finally J_v is executed in the interval $[D_1 - a_v, D_1]$;
- M_2 starts at time zero with J_v, then executes the jobs in S, and finally the remaining jobs.

This schedule is feasible, as $a_v + b_v \leq D_1$, since $J_v \notin S$. If such a job J_v does not exist, then we construct the following, currently infeasible schedule. On M_1, we first process the jobs from S and then the other jobs in arbitrary order in the interval $[0, \sum a_j]$; on M_2 we process the jobs in the same order as on M_1 in the interval $[a(S), a(S) + \sum b_j]$. If $a(S) + \sum b_j \leq D_2$, then we have discovered a feasible schedule, since $a(S) + b(S) > D_1$, which implies that the jobs not in S do not overlap in their execution. If $a(S) + \sum b_j > D_2$, then we check whether there exists a job J_s that is started before time D_2 and completed after time D_2. If there is no such job J_s, then we move the part scheduled in $[D_2, a(S) + \sum b_j]$ to the interval $[0, a(S)]$ on M_2. Since $D_2 \geq \sum b_j$, we know that this fits, and hence we are done, as there is no overlap. Suppose that we face the unlucky event that J_s does exist. We denote the set of jobs that are scheduled in between S and J_s by E and the jobs that succeed J_s by T. We adjust our schedule on M_2 by moving J_s and the jobs in T forward in front of the jobs in S, such that J_s starts at time zero and is followed by the jobs in T, whose relative order remains unchanged. If necessary, the jobs in S and E are shifted to the right. We call this schedule $\sigma(S)$; see Figure 1 for a schematic illustration. Since $a(S) > b_s$ (otherwise J_s would have qualified as the job J_v approximately twenty lines ago), the only jobs that may overlap are the jobs in T. Obviously, this can only be the case if the last job in T is completed on M_2 after the first job in T is started on M_1, that is, $b_s + b(T) > a(S) + a(E) + a_s$. Hence, if the current schedule is not feasible, then we must have that $a(S) + a_s < \sum b_j - b(S)$, from which we deduce that $a_s + a(S) + b(S) < D_2$. Since $b_s < a(S)$, we find that the makespan of the optimum schedule for $F2||C_{\max}$ applied to the job set $\{J_s\} \cup S$ is smaller than D_2. Hence, we can augment S with J_s and apply the same analysis again. This either leads to a feasible schedule meeting D_1 and D_2 or suggests a job that can be added to S. As the number of jobs is bounded, we will eventually find a feasible schedule that fits.

Theorem 3.1 *There exists a feasible schedule for the open shop problem in which M_1 and M_2 are finished at times D_1 and D_2 if and only if D_1 and D_2 satisfy the Conditions 1 – 4.* ∎

4 Finding the nondominated points (D_1, D_2)

Our example in the previous section shows that there can be at least two points (D_1, D_2) that lead to a feasible schedule, which are incomparable to each other. In this section we show that there are at most two such points and that they can be determined in $O(n)$ time. The properties of these D_1 and D_2 values depends on which of the lower bounds 1 – 4 is tight. We partition the instances in the following way, where as before $q = \arg\max_{1 \le j \le n}\{a_j + b_j\}$:

- $a_q + b_q \ge \max\{\sum a_j, \sum b_j\}$;
- $a_q + b_q \le \min\{\sum a_j, \sum b_j\}$;
- $\max\{\sum a_j, \sum b_j\} > a_q + b_q > \min\{\sum a_j, \sum b_j\}$.

In the first subcase, we have that $\max\{D_1, D_2\} = a_q + b_q$. Due to the Conditions 1 and 2, we arrive at the points $(\sum a_j, a_q + b_q)$ and $(a_q + b_q, \sum b_j)$. A fitting schedule is easily derived in both cases. In the first case, execute J_q in the intervals $[0, a_q]$ and $[a_q, a_q + b_q]$ on M_1 and M_2 and subsequently put the remaining operations in the gaps. A feasible schedule for $(a_q + b_q, \sum b_j)$ is derived in a similar fashion.

The second subcase is even easier: the point $(\sum a_j, \sum b_j)$ is clearly as small as possible in both components, and it is easily checked that in this case it satisfies Conditions 1 – 4. Hence, there is only one point of interest now.

In the third subcase, we assume without loss of generality that $\sum b_j > a_q + b_q > \sum a_j$. We first check whether $(\sum a_j, \sum b_j)$ satisfies the Conditions 1 – 4. If this is the case, then this is the only nondominated point. Otherwise, we must have that

$$\sum b_j < \min_{J_j \in S(\sum a_j)} a_j + \sum_{J_j \in S(\sum a_j)} b_j \equiv F^*(\sum a_j);$$

Hence, we find that the first nondominated point is $(\sum a_j, F^*(\sum a_j))$. To be able to arrive in another nondominated point, we must increase the D_1-value such that the set $S(D_1)$ decreases. The smallest D_1-value for which $S(D_1)$ changes is

$$\min_{J_j \in S(\sum a_j)} (a_j + b_j) \equiv Q.$$

But if D_1 is put equal to Q, then we can find a feasible schedule in which M_2 has finished all jobs at time $\sum b_j$ in the following way: If J_0 is the job that leads to Q, then execute J_0 in the intervals $[0, b_0]$ and $[b_0, a_0 + b_0]$ on M_2 and M_1, and put the remaining jobs in the gaps. Since the minimum D_2-value has been determined, there is no need to increase the D_1-value any further, which implies that $(\sum a_j, F^*(\sum a_j))$ and $(Q, \sum b_j)$ are the only nondominated points for the third subcase if $F^*(\sum a_j) > \sum b_j$.

References

[1] T. Gonzalez and S. Sahni (1976). Open shop scheduling to minimize finish time. *Journal of the Association for Computing Machinery 25*, 92-101.

[2] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan [1979]. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics 5*, 287–326.

[3] S.M. Johnson (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly 1*, 61-68.