Finding the Pareto Set for a Bi-criteria Scheduling Problem on a Single Machine with Equal Processing Times[1]

Alexander A. Lazarev

V.A. Trapeznikov Institute of Control Sciences of RAS, Moscow, Russian Federation;
Lomonosov Moscow State University, Moscow, Russian Federation;
Moscow Institute of Physics and Technology, Dolgoprudny, Russian Federation;
National Research University Higher School of Economics,Moscow, Russian Federation
e-mail: jobmath@mail.ru

Dmitry I. Arkhipov

Institute of Control Sciences of RAS, Moscow, Russia
e-mail: miptrafter@gmail.com

Frank Werner

Faculty of Mathematics, Institute of Mathematical Optimization, Otto-von-Guericke University Magdeburg, Germany
email: frank.werner@ovgu.de

January 7, 2015

Abstract: The following special case of the classical NP-hard scheduling problem $1|r_j|L_{max}$ is considered. There is a set of jobs $N = \{1, 2, \ldots, n\}$ with identical processing times $p_j = p$ for all jobs $j \in N$. All jobs have to be processed on a single machine. The optimization criterion is the minimization of maximum lateness L_{max}. We analyze algorithms for the makespan problem $1|r_j|C_{max}$, presented by Garey et al. (1981) and Simons (1978) and give two polynomial algorithms to solve the problem under consideration and to construct the Pareto set with respect to the criteria L_{max} and C_{max}. The complexity of the presented algorithms is $O(Q \cdot n \log n)$ and $O(n^2 \log n)$, respectively, where 10^{-Q} is the accuracy of the input-output parameters.

Keywords: scheduling, single machine, equal processing times, polynomial algorithms, bi-criteria problem

MSC: 90B35, 90C29

[1]Supported by RFBR-RZD grant 13-08-13190

1 Introduction

In this paper, the following scheduling problem is considered. There is a set of jobs $N = \{1, 2, \ldots, n\}$ and a single machine to execute jobs from this set. For each job $j \in N$, a release date r_j and a due date d_j are given. The processing time p is the same for all jobs of the set N. We define a *schedule* (or sequence) π as the execution sequence $K_1(\pi), K_2(\pi), \ldots, K_n(\pi)$, where

$$K_1(\pi) \cup K_2(\pi) \cup \cdots \cup K_n(\pi) \equiv N.$$

The equality $K_i(\pi) = j$ means that job $j \in N$ is executed under the ordinal number i in the schedule π. The execution of the job $K_i(\pi) = j$ starts at time

$$R_j(\pi) = \max\{C_{K_{i-1}}(\pi), r_{K_i(\pi)}\}$$

(where $C_{K_0}(\pi) = 0$) and finishes at time

$$R_j(\pi) + p = C_j(\pi),$$

where $C_j(\pi)$ is the *completion time* of the job $j \in N$. Let us denote the *lateness* of job j under the schedule π as

$$L_j(\pi) = C_j(\pi) - d_j.$$

The maximum completion time and the maximum lateness are denoted as $C_{\max}$ and $L_{\max}$, respectively. Let us call the schedule π *allowable* for the set N if all jobs according to the schedule π are executed without preemptions and intersections. We denote the set of all allowable schedules as Π. The goal is to find an allowable schedule $\pi \in \Pi$, which satisfies the following optimization criterion:

$$\min_{\pi \in \Pi} \max_{j \in N} L_j(\pi).$$

This problem $1|r_j, p_j = p|L_{max}$ is a special case of the classical NP-hard scheduling problem $1|r_j|L_{max}$. For the special case of the bi-criteria problem $1|r_j|L_{max}, C_{max}$ with the conditions

$$\begin{cases} d_1 \le \cdots \le d_n \\ d_1 - p_1 - r_1 \ge \cdots \ge d_n - p_n - r_n \end{cases}$$

Lazarev [1] has constructed a polynomial algorithm of complexity $O(n^3 \log n)$ for finding the Pareto set. It has been shown that this set contains at most n schedules. We note that a recent survey for solving parallel and single machine scheduling problems with equal processing times was given in [2]. In this paper, we consider some approaches to obtain an optimal solution for this special case and a related bi-criteria problem in polynomial time.

The remainder of the paper is as follows. In Section 2, we briefly discuss the trisection method applied in this paper. In Section 3, we discuss an auxiliary problem: the minimization of the maximum completion time subject to the constraint that the maximum lateness is bounded by a given constant. The main problem is considered in Section 4. An optimal solution is determined for the single machine problem with release dates for the bi-criteria problem of minimizing maximum lateness and the makespan.

2 Trisection search method

A simple way to obtain an optimal solution is the trisection method. In the first step, we find boundary values on the objective function L_{max}. For each job $j \in N$, we have:

$$d_j + L_j = C_j \ge r_j + p.$$

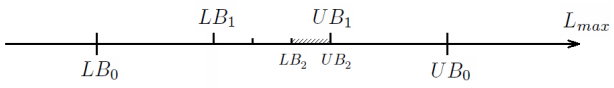

Figure 1: An example of the use of the trisection method.

Hence, a lower bound LB_0 on the optimal value L_{max} is as follows:

$$LB_0 = \min_{j \in N} \{r_j - d_j\} + p.$$

An upper bound UB_0 on the optimal value L_{max} can be given as follows:

$$UB_0 = L_{max}(\pi_C),$$

where π_C is an optimal schedule for the problem $1|r_j, p_j = p|C_{max}$. The fastest algorithm for solving this problem was presented in [3]. Let us use this algorithm to construct a schedule π_C with $O(n \log n)$ operations.

After finding the bounds LB and UB, we apply the trisection search method to find a solution of the problem as follows. In the first step, we divide the interval $[LB_0, UB_0]$ into three subintervals. Then, we set deadlines for all jobs $j \in N$ as follows:

$$D_j^1 = d_j + \frac{2}{3}LB_0 + \frac{1}{3}UB_0,$$
$$D_j^2 = d_j + \frac{1}{3}LB_0 + \frac{2}{3}UB_0.$$

Then we use the algorithm presented in [3] to construct the schedules π_C^1 and π_C^2 for the problems $1|r_j, D_j^1, p_j = p|C_{max}$ and $1|r_j, D_j^2, p_j = p|C_{max}$, respectively. If the schedule π_C^1 exists, set:

$$\begin{cases} LB_1 := LB_0 \\ \\ UB_1 := \frac{2LB_0 + 1UB_0}{3} \end{cases}$$

If the schedule π_C^1 does not exist and the schedule π_C^2 exists, we set:

$$\begin{cases} LB_1 := \frac{2LB_0 + 1UB_0}{3} \\ \\ UB_1 := \frac{1LB_0 + 2UB_0}{3} \end{cases}$$

Otherwise, we set:

$$\begin{cases} LB_1 := \frac{1LB_0 + 2UB_0}{3} \\ \\ UB_1 := UB_0 \end{cases}$$

This procedure is repeated until the difference $UB - LB$ is not larger than the accuracy 10^{-Q} of the input-output parameters. An example for the application of the trisection method is illustrated in Figure 1.

The number of necessary steps is equal to

$$\log_3[(UB_0 - LB_0) \cdot 10^Q].$$

In each step, the two schedules π_C^1 and π_C^2 are constructed. Hence, the total complexity is given by:

$$\log_3 \left[(UB_0 - LB_0) \cdot 10^Q\right] \cdot 2O(n \log n) = O(Q \cdot n \log n).$$

3 The auxiliary problem

3.1 Formulation of the auxiliary problem

Now we consider a second approach. We have to formulate an auxiliary problem to construct the second algorithm. We consider the same set of jobs $N = \{1, 2, \ldots, n\}$ and a *bound on the maximum lateness* y. The goal is to construct a schedule satisfying the following optimization criterion:

$$\min_{\pi \in \Pi} \max_{j \in N} C_j(\pi) | L_{\max}(\pi) < y.$$

For each set of due dates $d_1, \ldots, d_n$ and the bound on the lateness y, *deadlines* D_j can be calculated by the following formula:

$$D_j = d_j + y.$$

The auxiliary problem is the same as problem $1|r_j, p_j = p|C_{\max}$, but with one exception: the completion time C_j of the job $j \in N$ may not exceed its deadline D_j, i.e., we must have:

$$C_j < D_j.$$

An allowable schedule satisfying this restriction is called *feasible*. To construct the solution of the auxiliary problem, we consider the approach presented in [4]. Next, we briefly recall the main idea and the important notations from this paper.

The algorithm works as follows. While the completion times of all jobs are not larger than its deadlines, schedule the jobs according to the algorithm, presented in [5]. If for any job $X \in N$, the inequality

$$C_X \geq D_X$$

holds, then execute the special procedure $CRISIS(X)$ described in Section 3.2. This procedure finds the job A, which is already scheduled with the latest completion time, but for which

$$D_A > D_X$$

holds. This job is called $Pull(X)$, the time moment when its execution starts is denoted as $t_{A,X}$ and all jobs which are already scheduled after $Pull(X)$ and X constitute the *restricted set* $S(A, X]$ (see Fig. 2). The set of jobs, which belong to $S(A, X]$ and do not belong to any restricted subset of $S(A, X]$ is denoted as $\overline{S}(A, X]$. If $S(A, X]$ is a restricted set which is properly contained in $S' \subseteq N$, then $S(B, Z]$ is called an *inner* restricted set of S'. A *first-level* restricted set of S' is an inner restricted set of S' which is not contained in any other inner restricted set of S'. Simons [4] has shown that the number of jobs of $\overline{S}(A, X]$ which must be scheduled before the initial first-level restricted set of S(A,X], between any two first-level restricted sets, or after the final first-level restricted set is invariant over all possible schedules. We refer to the intervals in the schedule in which the jobs of $\overline{S}(A, X]$ must be scheduled as *first-level intervals* of $S(A, X]$. The procedure $CRISIS(X)$ reschedules the jobs of the set $\{A\} \cup S(A, X]$. The procedure fails when a job $Pull(X)$ for a crisis job X does not exist. After a successful execution of the procedure $CRISIS(X)$, Schrage's algorithm [5] is used to schedule the jobs. Such a scheduling is repeated until any call of the procedure $CRISIS()$ fails or all jobs from the set N have been successfully scheduled.

3.2 Procedures for the auxiliary algorithm

This algorithm consists of the following procedures: Schrage's algorithm [5]as well as the procedures $CRISIS(X)$ and $INVASION(S(C, W], r_{S(C,W]})$ from [4]. They are as follows.

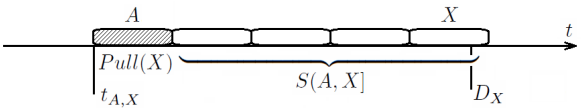

Figure 2: Job X experiences a crisis.

SCHRAGE's ALGORITHM:

1. Find the earliest release time:
   $t := \min\limits_{j \in N} r_j$.

2. Find a non-processed job, which is released at the moment t:
   $i := \arg \min\limits_{j: r_j \leq t} D_j$.

3. Process the job i, add this job to π and remove it from N:
   $C_i := t + p$;
   $\pi := \{\pi, i\}$;
   $N := N \setminus \{i\}$;

3. If $N \neq \emptyset$, then:
   set the time $t := \max \left\{ C_i, \min\limits_{j \in N} r_i \right\}$) and go to step 2.

4. Else:
   $return(\pi)$.

$CRISIS(X)$

1. Assume that the job X belongs to a minimal restricted set S'. If the job X does not belong to any restricted set, then $S' \equiv N$.
   Backtrack over the first-level jobs of S' looking for a job $Pull(X)$. Let $A = Pull(X)$ and define $S(A, X]$ to be a restricted set. If no job $Pull(X)$ exists, report a failure and stop.

2. Count the number of jobs of the set $\overline{S}(A, X]$ in each first-level interval of $S(A, X]$. Increase the counter of the initial first-level interval by 1.

3. Remove the jobs $\overline{S}(A, X]$ from the schedule.

4. $i := 1$.

5. While the required number of jobs of the set $S(A, X]$ has not been scheduled in the i^{th} first-level interval, schedule the jobs of the set $S(A, X]$ using Schrage's algorithm. (If $i = 1$, the first-level interval begins at time $r_{S(A,X]}$; otherwise, the interval begins at the time at which the preceding restricted set is completed).

   a) If some job Z experiences a crisis, call the procedure $CRISIS(Z)$.

   b) If some job Y overlaps with the restricted set $S(C, W]$, set $r_{S(C,W]}$ to be the time at which job Y is completed and call the procedure $INVASION(S(C, W], r_{S(C,W]})$.

6. If all jobs of the set $S(A, X]$ have been scheduled, then return;
   otherwise $i := i + 1$.

7. Go to step 5.

$INVASION(S(C,W], r_{S(C,W]})$

1. Count the number of jobs of the set $S(A,X]$ in each first-level interval of the set $S(A,X]$.
2. Steps 2-6 are identical to steps 3-7 of the subroutine CRISIS.

3.3 Algorithm for the auxiliary problem

The solution of the auxiliary problem $1|r_j, p_j = p, D_j|C_{max}$, presented in [4], is as follows. Find the schedule π by means of Schrage's algorithm and use then the auxiliary algorithm.

AUXILIARY ALGORITHM

1. While the set N has not been completely checked under the schedule π,
   check whether for all jobs $j \in N$ the inequality $C_j(\pi) < D_j$ holds;
   otherwise stop (the set N has been successfully scheduled).
   a) If some job X experiences a crisis, call the procedure $CRISIS(X)$.
   b) Else $return(\pi)$.

The two following theorems from [4] can be used.

Theorem 1 *After the execution of the auxiliary algorithm, an optimal set with respect to the criterion C_{max} is constructed, provided that the schedule π is constructed by Schrage's algorithm.*

The proof of this theorem was given in [4].

Theorem 2 *Let $S(A,X]$ be a restricted set. If a feasible schedule exists, then the assertions 1-4 hold for all feasible schedules. Each time a procedure $CRISIS()$ or $INVASION()$ is used to schedule the jobs of the set $S(A,X]$, the following assertions 1-3 hold:*

1. *The first job of the set $S(A,X]$ is always scheduled to start in the interval $(t_{A,X}, t_{A,X} + p)$,*
2. *Only jobs from the set $S(A,X]$ can be scheduled completely in the interval $(t_{A,X}, D_x]$.*
3. *The set $S(A,X]$ cannot be scheduled such that a job starts before time $r_{S(A,X]}$.*

   When the program returns from a call of procedure $CRISIS()$ or $INVASION()$, the following assertion holds:

4. *The set $S(A,X]$ cannot be completed at an earlier time than the time at which it is currently scheduled to be completed.*

The proof of this theorem was given in [4].

4 Solution of the main problem

Next, we consider the main problem $1|r_j, p_j = p|L_{max}$. We present an algorithm to obtain the Pareto set of schedules with respect to the criteria L_{max} and C_{max}. First, we introduce a procedure $CHECK(\pi, N, y)$ which is as follows.

$CHECK(\pi, N, y)$

1. Set the bound y.
2. Set the deadlines $D_i := d_i + y$.
3. If all jobs from the set N have been scheduled, go to step 7.
4. While t is not in the interval $[r_{S(A,X]}, D_X)$ for any restricted set $S(A, X]$ from the schedule π that has not yet been completely performed, execute the jobs under π^* according to Schrage's algorithm.
5. Otherwise, execute only the jobs from the set $S(A, X]$ under the schedule π, and then go to step 3.
6. If in steps 4-5 any job Y experiences a crisis, run the procedure $CRISIS(Y)$.
7. $return(\pi^*)$.

Lemma 1 *Let π and π' be the schedules constructed by the auxiliary algorithm for the bounds y and y', respectively, and*

$$\pi^* = CHECK(\pi, N, y).$$

If $y < y'$, then

$$CHECK(\pi, N, y) = \pi'.$$

holds.

Proof: Assume the contrary. We compare the schedules $CHECK(\pi, N, y)$ and π' from $t = 0$ up to C_{max}. Suppose that at some moment t, the first difference was found. The possible cases of such a difference are illustrated in Figure 3, and they are as follows:

1) **Two different jobs start at the moment** t.

   This type of a difference is impossible, because both algorithms ensure that at the moment t only the job with a minimal deadline can start its execution.

2) **There is an execution of the job** X **under the schedule** π^* **and an idleness under the schedule** π' **at the moment** t.

   The job X is not executed in the sequence π' in spite of $r_X \leq t$. Hence, we have $X = Pull(Y)$ in the schedule π. According to the time $r_{S(X,Y]}$ and Theorem 2, the jobs from the set $S(X, Y]$ cannot finish their execution in the schedule π^* before the time $D_Y(\pi')$ because the first of them starts at time $t_{X,Y} + p$. Hence, some jobs from the set $S(X, Y]$ experience a crisis in the schedule π^*. In the last call of $CRISIS(Y')$, $Y' \in S(X, Y]$, we have $CRISIS(Y') = X$. Thus, the job X cannot start at the moment t.

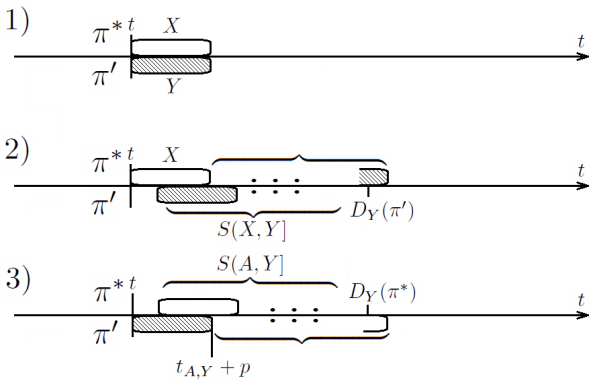

Figure 3: Cases of the possible difference of the schedules $CHECK(\pi, N, y)$ and π'.

3) **There is an execution of the job X under the schedule π' and an idleness under the schedule π^* at the moment t.**

The idleness is related to some restricted set $S(A, Y]$ from the set of jobs in π. The schedule π' is feasible for the set of jobs N and the deadlines $D_i(\pi^*)$ since $y < y'$. According to Theorem 2, the jobs from the set $S(A, Y]$ cannot start their execution under the schedule π' at the moment $t_{A,Y} + p$ and finish their execution until the moment $D_Y(\pi^*)$.

Hence, all above three cases are impossible and there are no differences between the schedules π^* and π'.

Q.E.D.

Next, we describe the main algorithm M to obtain the Pareto set with respect to the criteria L_{max} and C_{max}.

MAIN ALGORITHM (Algorithm M)

1. Set the bound $y_0 := +\infty$.
2. Construct the schedule π_1 according to the auxiliary algorithm,
   and add it to Φ, i.e.: $\Phi := \{\pi_1\}$;
   set the counter $k := 1$;
   set the bound $y_1 := L_{max}(\pi_1)$.
3. Construct the schedule $\pi_{k+1} = CHECK(\pi_k, N, y_k)$.
   a) If the schedule $CHECK(\pi_k, N, y)$ exists, then:
      add π_{k+1} to the set Φ, i.e.: $\Phi := \Phi \cup \pi_k$;
      set $y_k = L_{max}(\pi_k)$;
      increase the counter k, i.e.: $k := k + 1$;
      repeat step 3.
   b) Otherwise, $return(\Phi)$.

Lemma 2 *If any job becomes a crisis job for the second time, then the algorithm stops.*

Proof: When a restricted set is formed, all the jobs in this set have deadlines not greater than the deadline of the crisis job which triggered the restricted set. Consequently, if that job experiences a crisis for a second time, the algorithm will not find a job to pull and so there will be a failure in step 1 of the subroutine $CRISIS()$.

Q.E.D.

Theorem 3 *After the execution of Algorithm M, the Pareto set of schedules Φ, $|\Phi| \leq n+1$, according to the criteria L_{max} and C_{max} has been constructed, and the schedule π^* is an optimal solution for the main problem.*

Proof: When Algorithm M has terminated, the set of schedules Φ has been constructed. For each pair of consecutive schedules π_x, π_{x+1} of the set Φ, the inequalities

$$\begin{cases} L_{max}(\pi_{x+1}) < y_{x+1} \\ L_{max}(\pi_x) < y_x. \end{cases}$$

hold. Moreover, for these two schedules the inequality

$$C_{max}(\pi_{x+1}) \geq L_{max}(\pi_x)$$

also holds, because π_x is an optimal schedule with respect to the criterion $C_{max}|L_{max} < y_x$ and $y_{x+1} < y_x$. Hence, we have

$$L_{max}(\pi_{|\Phi|-1}) < \cdots < L_{max}(\pi_1) < L_{max}(\pi_0),$$

$$C_{max}(\pi_{|\Phi|-1}) \geq \cdots \geq C_{max}(\pi_1) \geq C_{max}(\pi_0).$$

This implies that the schedule $\pi^* = \pi_{|\Phi|-1}$ is an optimal schedule with respect to the criterion L_{max}. In each call of the procedure $CHECK()$ by Algorithm M, the subroutine $CRISIS()$ is executed at least once, because the schedules π_k and π_{k+1} are different. Hence, we get:

$$|\Phi| \leq n+1.$$

Q.E.D.

Lemma 3 *The complexity of Algorithm M is $O(n^2 \log n)$.*

Proof: According to Theorem 1, for each job $X \in N$, the procedure $CRISIS(X)$ is executed not more than once. Hence, the total number of running the procedure $CRISIS()$ is not more than n. During the procedure $CRISIS(X)$, each job from the set N is rescheduled not more than once. Hence, each job is scheduled not more than $n+1$ times by Schrage's algorithm during the construction of the schedules $\pi_1, \ldots, \pi_{|\Phi|}$, and not more than n times due to the execution of the procedure $CRISIS()$. Hence, the total number of reschedulings is not more than $(2n+1) \cdot n$. It is still necessary to multiply this result by the factor $\log n$ due to the use of the heaps. This leads to a total complexity of $O(n^2 \log n)$.

Q.E.D.

5 Concluding Remarks

In this paper, two approaches to solve the problem $1|r_j, p_j = p|L_{max}$ were presented. In addition, the Pareto set with respect to the criteria L_{max} and C_{max} was constructed. The efficiency of these approaches depends on the number of jobs and the accuracy of the input-output parameters. The core idea of the second approach was to construct a schedule with lower L_{max} value than in the previous step, but to use the knowledge obtained in the previous steps. This allowed us to adopt a makespan algorithm to the criterion L_{max} without a substantial increase of the complexity.

References

[1] Lazarev, A.A.: The Pareto-optimal set of the NP-hard problem of minimization of the maximum lateness for a single machine. *Journal of Computer and Systems Sciences International. M.: SP MAIK Nauka/Interperiodica.* 2006. 45, No. 6, 943 - 949.

[2] Kravchenko, S.A. and Werner, F.: Parallel Machine Problems with Equal Processing Times. *Journal of Scheduling.* Vol. 14, No. 5, 2011, 435 - 444.

[3] Garey, M.R.; Johnson, D.S.; Simons, B.B. and Tarjan, R.E.: Scheduling unit-time tasks with arbitrary release times and deadlines. *SIAM J. COMPUT.* Vol. 10, No. 2, May 1981, 256 - 269.

[4] Simons, B.B.: A fast algorithm for single processor scheduling. *In 19th Annual Symposium on Foundations of Computer Science (Ann Arbor, Mich., 1978)*, 246-252.

[5] Schrage, L.: Solving Resource-Constrained Network Problems by Implicit Enumeration: Non Preemptive Case. *Operations Research.* Vol. 18 Issue 2, 1970, 263 - 278.