Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 48-3 (2015) 806–809

Minimization of maximum lateness with equal processing times for single machine ⋆

Alexander A. Lazarev * **Dmitry I. Arkhipov** **

* *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.*

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

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

Keywords: scheduling, unit-time jobs, polynomial algorithms, dynamic programming

1. INTRODUCTION

1.1 Formulation of the main problem

The following problem of scheduling theory is considered. There is a set of jobs N and a single machine to process 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* π as an execution sequence $K_1(\pi), K_2(\pi), \dots, K_n(\pi)$, where $K_1(\pi) \cup K_2(\pi) \cup \dots \cup K_n(\pi) \equiv N$. The equality $K_i(\pi) = j$ means that job $j \in N$ is processed under the ordinal number i under the schedule π. The execution of the job $K_i(\pi) = j$ starts at time $R_i(\pi) = \max(C_{i-1}(\pi), r_{K_i(\pi)})$ and finishes at time $R_i(\pi) + p = C_j(\pi)$, where $C_j(\pi)$ is the *completion time* of the job $j \in N$. Let us denote *lateness* as $L_j(\pi) = C_j(\pi) - d_j$. The maximum completion time and 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 under the schedule π executed without preemptions and intersections. We denote the set of all allowable schedules as Π. The goal is to find allowable schedule $\pi \in \Pi$, which satisfies the following optimization criteria:

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

This problem $1|r_j, p_j = p|L_{max}$ is a special case of classical NP-hard scheduling problem $1|r_j|L_{max}$. Now, let us consider some approaches to obtain the solution in polynomial time.

2. DICHOTOMY METHOD

A simple way to obtain the solution is the dichotomy (trisection search) method. In the first step, we find boundary values on the objective function L_{max}. Each job $j \in N$ holds:

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

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 can be estimated as:

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

where π_C is an optimal schedule for the problem $1|r_j, p = const|C_{max}$. The fastest algorithm for solving this problem was presented by Garey et al. (1981). Let us use this algorithm to construct π_C in $O(n \log n)$ operations.

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

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

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

Then we use algorithm presented by Garey et al. (1981) to construct the optimal schedules π_C^1 and π_C^2 for the problems $1|r_j, d_j^1, p = const|C_{max}$ and $1|r_j, d_j^2, p = const|C_{max}$ respectively. If the schedule π_C^1 exists, set:

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

⋆ Supported by RFBR-RZD grant 13-08-13190

10.1016/j.ifacol.2015.06.182

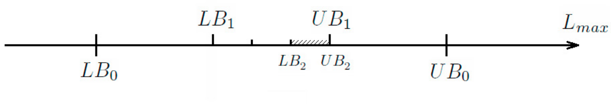

Fig. 1. An example of the use of three times dichotomy.

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

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

Otherwise, we set:

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

This procedure repeated until the difference $UB - LB$ is not larger than 10^{-Q} – the accuracy of the input-output parameters. An example of the dichotomy method usage illustrated in figure 1.

The number of steps is equal to

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

In each step the two schedules π_C^1, π_C^2 are constructed. Hence, the total complexity equals:

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

3. THE AUXILIARY PROBLEM

3.1 Formulation of auxiliary problem and algorithm

Let us consider the second approach. We have to formulate an auxiliary problem to construct the second algorithm. We consider the same set of jobs $N = \{1, \ldots, n\}$ and a *bound on the maximum lateness* y. The goal is to construct a schedule with respect to ctiteria:

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

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 $1|r_j, p = const|C_{\max}$, but with one exception: the completion time C_j of the job j may not exceed the deadline D_j:

$$C_j < D_j.$$

An allowable schedule satisfying this restriction is called *feasible*. To construct the solution of the auxiliary problem, we consider the approach presented by Simons (1978). 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 jobs according to algorithm, presented by Schrage (1970). If for any job $X \in N$ the inequality

$$C_X \geq D_X$$

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

$$D_A > D_X$$

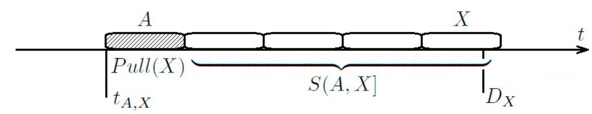

Fig. 2. Job X experiences crisis.

Schrage's Algorithm

1. Find earliest release time:
   $t = \min_{j \in N} r_j$.
2. Find a non-processed job, which released at the moment t:
   $i = \arg \min_{j: r_j \leq t} D_j$.
3. Process job i, add it 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 $t = \max(C_i, \min_{j \in N} r_i)$ and go to step 2.
4. Else:
   $return(\pi)$.

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* (r.s.) $S(A, X]$ (see fig. 2). The set of jobs, which belong to $S(A, X]$ and not belong to any restricted subset of $S(A, X]$ is denoted as $\overline{S}(A, X]$. The procedure $CRISIS(X)$ reschedule a set of jobs $\{A\} \cup S(A, X]$. The procedure fails when $Pull(X)$ for a crisis job X does not exist. After a successful execution of the procedure $CRISIS(X)$, Schrage's algorithm (see Schrage (1970)) is used to schedule the jobs. Such a scheduling repeats until any call of procedure $CRISIS()$ fails or all jobs from the set N have been successfully scheduled.

3.2 Procedures for auxiliary algorithm

This algorithm consists of the following procedures: the algorithm presented by Schrage (1970) as well as the procedures $CRISIS(X)$ and $INVASION(S(C, W], r_{S(C,W]})$ presented by Simons (1978).

3.3 Algorithm for auxiliary problem

The solution of auxiliary problem $1|r_j, p = const, D_j|C_{max}$, presented in Simons (1978) is as follows. Find the schedule π by means of Schrage's algorithm, and use then the auxiliary algorithm.

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.

Proof. The proof of this theorem is given in Simons (1978).

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

$CRISIS(X)$

1. Assume that X belongs to a minimal r.s. S'. If X does not belongs to any restricted set, then $S' \equiv N$. Backtrack over the first level jobs of S' looking for $Pull(X)$. Let $A = Pull(X)$ and define $S(A, X]$ to be a restricted set. If no $Pull(X)$ exists, report failure and halt.
2. Count the number of jobs of $\overline{S}(A, X]$ in each first level interval of $S(A, X]$. Increase the count 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 S(A,X] have not been scheduled in the i^{th} first level interval, schedule the jobs of $S(A, X]$ using the naive algorithm. (If $i = 1$, the first level interval begins at $r_{S(A,X]}$; otherwise, the interval begins at the time at which the preceding r.s. is completed).
   a) If some job Z has a crisis, call $CRISIS(Z)$.
   b) If some job Y invades the following r.s. $S(C, W]$, set $r_{S(C,W]}$ to be the time at which Y is completed and call $INVASION(S(C, W], r_{S(C,W]})$.
6. If all the jobs of $S(A, X]$ 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 $S(A, X]$ in each first level interval of $S(A, X]$.
2. Steps 2-6 are identical to steps 3-7 of the CRISIS subroutine.

AUXILIARY ALGORITHM

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

1. The first job of $S(A, X]$ is always scheduled to begin in $(t_{A,X}, t_{A,X} + p)$,
2. Only jobs in $S(A, X]$ can be scheduled totally in $(t_{A,X}, D_x)$.
3. $S(A, X]$ can not be scheduled to begin before $r_{S(A,X]}$.
   When the program returns from a procedure call the following assertion holds:
4. $S(A, X]$ can not be completed any earlier than the time at which it is currently scheduled to be completed.

Proof. The proof of this theorem is given in Simons (1978).

4. MAIN PROBLEM SOLUTION

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

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

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

$CHECK(\pi, N, y)$

1. Set the bound y.
2. Set deadlines $D_i = d_i + y$.
3. If all jobs from 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, execute the jobs under π^* according to Schrage's algorithm.
5. Otherwise, execute under π only the jobs from the set $S(A, X]$, and then go to step 3.
6. If in steps 4-5 any job Y experiences a crisis, run $CRISIS(Y)$.
7. $return(\pi^*)$.

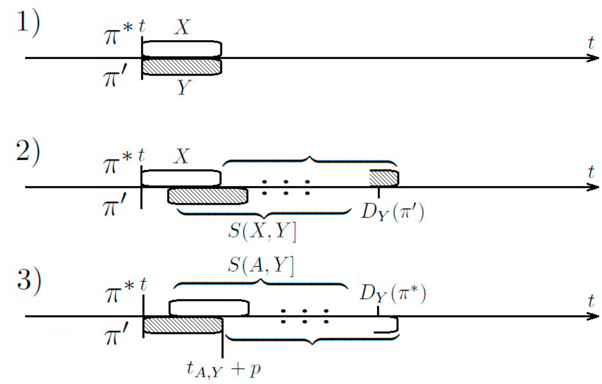

Fig. 3. Cases of the possible difference.

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 the moment t the first difference was found. The possible cases of the difference illustrated in figure 3, and they are as follows:

1) **Two different jobs start at the moment t.**
   This type of difference is impossible, because both algorithms ensure that at the moment t only the job with a minimal due date 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 $r_X \le t$. Hence, $X = Pull(Y)$ under 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 $S(X, Y]$ experience a crisis under 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 starts at the moment t.
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 $S(A, Y]$ can not start their execution under the

MAIN ALGORITHM

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 $k := k + 1$;
      repeat step 3.
   b) Otherwise, $return(\Phi)$.

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 π^* and π'.
Q.E.D.

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

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

Proof. When a r.s. is formed, all the jobs in this set have deadlines no greater than the deadline of the crisis job which triggered the r.s. Consequently, if that job experiences a crisis for a second time, the algorithm will not find a job to pull and will be a failure in step 1 of the subroutine $CRISIS()$.

Theorem 3. After the execution of algorithm 4, 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 4 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 this two schedules the inequality

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

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

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

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

This implies that the schedule $\pi^* = \pi_{|\Phi|-1}$ is an optimal according to the criterion L_{max}. On each $CHECK()$ procedure of algorithm 4 $CRISIS()$ subroutine executes at least once, because the schedules π_k and π_{k+1} are different. Hence, we get

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

Lemma 3. The complexity of Algorithm 4 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 $CRISIS()$ is not more than n. During the procedure $CRISIS(X)$ each job from the set N rescheduled not more than once. Hence, each job schedules not more than $n+1$ times by Schrage's algorithm during the construction of sets $\pi_1, \ldots, \pi_{|\Phi|}$, and not more than n times due to execution of $CRISIS()$ procedures. Hence, total number of reschedulings is not more than $(2n + 1) \cdot n$. It is still necessary to multiply this result on $\log n$ due to the use of the heaps. This leads to a total complexity of $O(n^2 \log n)$.

5. CONCLUSION

Two approaches to solve the problem $1|r_j, p = const|L_{max}$ are 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 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.

ACKNOWLEDGEMENTS

Authors would like to thank Prof. Frank Werner and Dr. Ruslan Sadykov for their active assistance in working on this problem.

REFERENCES

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

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

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