Approximation Algorithms for Single Machine Scheduling Problem with Delayed Precedence Constraints

Natalia Grigoreva
Saint Petersburg State University
Saint Petersburg, Russia
e-mail: n.s.grig@gmail.com

Alexey Didenko
ITMO National Research University
Saint Petersburg, Russia
e-mail: alex.d.t.610@gmail.com

***Abstract*—This paper presents a new approximation algorithm for the single machine scheduling problem with release times and delivery times, which includes additional conditions that require a certain delay between the completion of one task and the start of the next (delayed precedence constraints).**

A computational experiment has demonstrated that the proposed algorithm effectively creates schedules, and when combined with the well-known Jackson algorithm, it can enhance the quality of the final schedule.

***Keywords*— Single machine scheduling, deferred precedence constraints, approximations algorithm.**

I. Introduction

This paper is concerned with a single machine scheduling problem of minimizing the makespan. The problem relates to the scheduling problem and has many applications. We consider a system of jobs $U = \{u_1, u_2, \ldots, u_n\}$. Each job is characterized by its execution time $t(u_i)$, its release time (or head) $r(u_i)$, and its delivery time (or tail) $q(u_i)$. Precedence constructions between jobs are represented by a directed acyclic task graph $G = \langle U, E \rangle$. E is a set of directed arcs, an arc $e = (u_i, u_j) \in E$ if and only if $u_i \prec u_j$.

The expression $u_i \prec u_j$ means that the job u_j may be initiated only after completion of the job u_i. If $u_i \prec u_j$, we call job u_i a predecessor of job u_j and job u_j a successor of job u_i.

Release time $r(u_i)$ is the time at which the job is ready to start processing, and its delivery begins immediately after processing has been completed.

At most one job can be processed at a time, but all jobs may be simultaneously delivered. The set of jobs is performed on a single processor. Job preemption is not allowed.

The schedule defines the start time $\tau(u_i)$ of each job $u_i \in U$. For each arc $e = (u_i, u_j) \in E$ we define delay $l_{ij} \geq t(u_i)$ such as $\tau(i) + l_{ij} \leq \tau(j)$. This constraint is called the deferred precedence constraint (DPC).

The makespan of the schedule S is the quantity

$$C_{\max} = \max\{\tau(u_i) + t(u_i) + q(u_i) | u_i \in U\}.$$

The objective is to minimize $C_{\max}$, the time by which all jobs are delivered.

Following the 3-field classification scheme proposed by Graham *et al.* [8], the problem under consideration is denoted by $1|r_j, q_j, dpc|C_{\max}$.

If $l_{ij} = t(u_i)$, the problem is equivalent $1|r_j, q_j, prec|C_{\max}$.

The problem $1|r_j, q_j, dpc|C_{\max}$ is a more accurate representation of the problem of planning on a single machine in a multi -machine system, such as planning on the shop floor [7], than $1|r_j, q_j|C_{\max}$.

The problem is a generalization of a known problem for a single processor $1|r_i, q_i|C_{\max}$. In this problem, there is no precedence construction between jobs, and all jobs are independent. Each job is characterized by its execution time $t(u_i)$, its release time $r(u_i)$, and its delivery time $q(u_i)$. This problem has been studied well, and several algorithms have been developed for constructing approximate solutions. Lenstra in [13] showed that the problem is NP-hard in the strong sense.

The first algorithm for constructing an approximate schedule was the Schrage heuristic, an extended Jackson rule, which is formulated as follows: each time the processor is released, a ready job with the maximum delivery time is assigned to it [19].

The computational complexity of the algorithm is $O(n \log n)$. Kize et al. [12] showed that the algorithm has a guaranteed accuracy score of 2. Potts [18] proposed an $O(n^2 \log n)$ iterated algorithm, Nowicki and Smutnicki [16] presented a more efficient 3/2 approximation algorithm, which runs in $O(n \log n)$. Hall and Shmoys [11] improved the Potts algorithm when they applied it to both the original and reversed problems (in which the release date and delivery times are reversed). The worst-case performance ratio of the algorithm is equal 4/3. All the mentioned algorithms use the list-based greedy Schrage algorithm as a basic heuristic.

The author proposed an approximate algorithm IJR that allows the processor to stand idle while waiting for higher-priority job, which runs in $O(n \log n)$. In [9], we proved that the worst-case performance ratio of the IJR algorithm is equal 11/7. The combination of the two algorithms, the IJR algorithm and Schrage algorithm, allowed us to improve the

https://doi.org/10.51408/csit2025_60

performance ratio to 3/2 [10].

The works of Baker and Su [3], McMahon and Florian [15], Carlier [4], Grabowski et al. [6], Pan and Shi [17] developed branch and bound algorithms to solve the problem without and with precedence constraints using different branching rules and bounding techniques. The most efficient algorithm is the algorithm by Carlier, which optimally solves instances with up to thousands of jobs. This algorithm constructs a full solution in each node of the search tree. Liu [14] modified the Carlier algorithm to solve the problem with precedence constraints very efficiently.

The $1|r_j, q_j, prec|C_{\max}$ is a key component of several more complex scheduling problems. Problem is useful in solving flowshop and jobshop scheduling problems [1], [5] and plays a central role in some real industrial applications [20].

It is impossible to apply the Carlier algorithm to the problem $1|r_j, q_j, dpc|C_{\max}$ with delays. Balas et.al. [2] proposed the branch and bound algorithm with a new branching scheme. The article [21] proposed a modification of the largest tail heuristic (LTH) and an alternative search strategy in the search tree for exploring the branch and bound method.

In this paper, we propose approximate scheduling algorithms for the deferred precedence constraint problem. The main idea behind the algorithm is to allow the processor to remain idle until a higher-priority task becomes available. The conditions under which it is justified to use an idle processor are determined.

The article is organized as follows. The adapted Jackson algorithm for the delay problem is briefly described in Section 2. The third section describes a new algorithm for the delay problem. The results of the computational experiment are briefly described in 4 sections.

II. Jackson's Algorithm

First, we present a modification of the Jackson algorithm for solving the problem with delays. To find a new release time for a job u, you need to find the length of the maximum path in the graph $G = \langle U, E\rangle$ from the starting vertex to the job u. To find a new delivery time, you need to build the maximum path from the job u to the final vertex.

If $u_i \prec u_j$ then we change $r(u_j) = \max\{r(u_j), r(u_i) + l(i,j)\}$ and $q(u_i) = \max\{q(u_i), q(u_j)+l(i,j)\}$. This replacement does not affect the feasibility of any schedule.

We store ready jobs in a queue with priorities Q_1; the priority is the delivery time. Let $time$ be the time when the processor is released after completing the tasks already scheduled. At the initial time, we assume $time := r_{\min}$. A partial schedule S_k is a sequence of tasks installed on the processor over k iterations.

The pseudocode of the extended Jackson rule method for constructing the approximation schedule is shown in the algorithm 1.

III. ILTF Algorithm

In this section, we propose a new algorithm ILTF to solve the problem with delays. This algorithm is a modification of the algorithm proposed by the author in [9], which has a guaranteed accuracy score of 11/7 for the problem $1|r_j, q_j, dpc|C_{\max}$.

Algorithm 1 LTH algorithm

```
time := r_min; k := 1; S_0 := ∅, Q_1 := ∅.
while k ≤ n do
    We add ready tasks to the queue Q_1such as r(i) ≤ time.
    if Q_1 = ∅ then
        time := min{r(i)|i ∉ S_{k-1}}
    end if
    Selest ready job u such as q(u) = max{q(i)|i ∈ Q_1}.
    we install the task u on the processor S_k = S_{k-1} ∪ u
    set τ(u) := time;
    time := τ(u) + t(u);
    recalculate r(j) for all successors j of job u. ; If u ≺ j
    then we change r(j) = max{r(j), r(u) + l(u, j)}
    k := k + 1
end while
```

We define the lower bound of the objective function. The precedence constraints are given by an acyclic graph $G(V, E)$. Determine the length of arc $e = (u_i, u_j) \in E$ $d(e) = l_{ij}$. Find a possible idle time of the processor due to delays $D = \sum_{e\in E}(d(e) - t(u_i))$. We add two vertices to the graph: s is the beginning vertex and t is the ending vertex .

We introduce a set of arcs $E_s = \{e_{si}\}$. Each arc $e_{si} = (s, u_i)$ connects the initial vertex s with a vertex u_i that has no predecessors (let's denote a set of such jobs U_s). Determine the length of this arc $d(e_{si}) = r(u_i)$ and $r_{\min} = \min\{r(u_i) \| \ u_i \in U_s\}$.

We add a set of arcs $E_t = \{e_{it}\}$. Each arc$e_{it} = (u_i, t)$ connects a vertex u_i that has no descendants (let's denote a set of such jobs U_t) the final vertex t. Determine the length of this arc $d(e_{si}) = t(u_i)+q(u_i)$ and define $q_{\min} = \min\{q(u_i) \| \ u_i \in U_t\}$.

We can determine the length of the critical path t_{cp} in this graph $G(V\cup\{s,t\}, E\cup E_s\cup E_t)$. The length of the critical path allows us to refine the lower bound of the objective function $LB1 = \max\{t_{cp}, \max\{r(u_i) + t(u_i) + q(u_i) \mid u_i \in V\}\}$. $LB2 = r_{\min} + \sum_{i=1}^{n} t(u_i) + q_{\min}$. $LB = \max\{LB1, LB2\}$. When constructing a critical path, we will get the length of the maximum path $p(u_i)$ from the initial vertex s to the vertex u_i. $p(u_i)$ sets the new release time $r^*(u_i) = \max\{r(u_i), p(u_i)\}$. To find a new delivery time $q(u_i)$, we need to build the maximum path from the job u_i to the final vertex, the length $w(u_i)$. Then $q^*(u_i) = \max\{q(u_i), w(u_i)\}$.

Let S_k be the set of jobs that have been scheduled after k iterations. Algorithm ILTF is a greedy algorithm in the sense that at each iteration, it adds any ready job to S_k.

The task to be assigned to the processor is selected in three stages: first, the ready task u with the maximum delivery time is selected. Then a task u^* is selected that can be started before the end of the task u and $q(u^*) > LB/2$ and $q(u^*) - q(u) > r(u^*) - time$. If there is no such task, then we put task u on the processor.

If the job u^* exists, then we are looking for a job u_1 that can be done during processor downtime from the moment $time$ before the start of job $r(u^*)$. If the job u_1 has been found, then we assign it to the processor, otherwise, we assign u^*.

Algorithm ILTF generates the schedule in which processor is kept idle at a time when it could begin processing a job.

Definition 3.1: A job $u \notin S_k$ is called the ready job at the level k, if all its predecessors are included in the partial solution S_k.

Let $time$ be the completion time of the last completed job in S_k.

Then $time := \max\{\tau(u) + t(u) \mid u \in S_k\}$. Q_k - a set of ready jobs at the level k

The pseudocode of the ILTF algorithm is shown in the algorithm 2.

A. Combined scheduling algorithm ICAD

1. Build a schedule S_{LTH} using Jackson's algorithm. The makespan of the schedule is $C_{\max}(S_{LTH})$.

21. Build a schedule S_{ILTF} using the ILTH algorithm. The makespan of the schedule is $C_{\max}(S_{ILTF})$.

3. Choose the schedule S_A with a smaller value of the objective function:
$C_{\max}(S_A) = \min\{C_{\max}(S_{ILTH}), C_{\max}(S_{LTH})\}$.

IV. Computational Experiment

A computational experiment was conducted to compare the performance of LTH and ILTF algorithms for the problem of the single processor scheduling with delays. We investigate instances with precedence constraints and delays. The instance generation scheme is as follows: for $1 \le i < j \le n$ a precedence constraint $i \prec j$ is generated when $\gamma < P_{ij}$, where γ is generated from the uniform distribution over the interval [0,1], and $D = 0, 0.1, 0.3, \ldots, 0.9$.

$$P_{ij} = \frac{D(1-D)^{(j-i-1)}}{1 - D(1-(1-D)^{(j-i-1)})}.$$

Job processing times are generated with discrete uniform distributions between 1 and 50. Release dates, delivery times, and delays $l_{i,j}$ are generated with discrete uniform distributions between 1 and Kn, for $K = 20$.

If $l_{i,j} < t(i)$ then $l_{i,j} := l_{i,j} + t(i)$.

Parameter K controls the range of heads and tails. For each n and K, we generate 100 instances.

For tests of type A, the number of jobs n was changed from 50 to 5000, and for all tests, the value $K = 20$ was chosen. The value of the objective function $C\max$ was compared with the lower bound LB. The average relative error $R = C_{\max}/LB$ of the solution decreases with increasing n for all algorithms. The average relative error is 8 percent for the ILTF algorithm, 9 percent for the LTH algorithm and 4 percent for the ICAD algorithm (for n = 50).

A computational experiment was conducted for various types of tests based on comparing the total delay time with the total task completion time. Let

$$H = \frac{\sum_{i=1}^{n} t(u_i)}{\sum_{e \in E}(d(e) - t(u_i))}.$$

Algorithm 2 ILTF algorithm

Calculate the new release time $r(u_i)$
Calculate the length of the critical path t_{cp}
Calculate new delivery time $q(u_i)$
$r_{\min} = \min\{r(u) \mid u \in U_s\}$,
$q_{\min} = \min\{q(u) \mid u \in U_t\}$
Define the lower bound LB of the optimal makespan
$LB = \max\{t_{cp}, r_{\min} + \sum_{i=1}^{n} t(u_i) + q_{\min}\}$
$LB = \max\{LB, \max\{r(u_i) + t(u_i) + q(u_i) \mid u_i \in V\}\}$.
Q_k is the set of ready jobs
$time := r_{\min}; k := 0; S_0 := \emptyset, Q_0 := \emptyset$.
if $r(u_i) = r_{\min}$ **then**
  $u_i -> Q_0$
end if
The main cycle
while $k \le n$ **do**
  $r := \min\{r(u_i) \mid u_i \in Q_k\}$
  if $r > time$ **then**
    $time := r$
  end if
  Select the ready job $u \in Q_k$, such as
  $q(u) = \max\{q(u_i) \mid r(u_i) \le time\}$.
  Select the ready job u^*, such as
  $q(u^*) = \max\{q(u_i) \mid time < r(u_i) < time + t(u)\}$
  if $q(u^*) > LB/2$ & $q(u^*) - q(u) > r(u^*) - time$ **then**
    Select the ready job u_1 such as
    $q(u_1) = \max\{q(u_i) \mid r(u_i) + t(u_i) \le r(u^*)\}$.
    if we find u_1 **then**
      select job $v = u_1$.
    else
      $v = u^*$.
    end if
  else
    $v = u$
  end if{ Define the start time of job v}
  $\tau(v) := \max\{time, r(v)\}$;
  $time := \tau(v) + t(v)$;
  recalculate $r(j)$ for all successors of job v ;
  $S_{k+1} = S_k \cup v$
  $k := k + 1$
end while
We construct the approximation schedule $S = S_n$
$C_{\max}(S) = \max\{\tau(u_i) + t(u_i) + q(u_i) \mid u_i \in U\}$.

Examples with one large job u, such as $t(u) > 1/2 \sum_{i=1}^{n} t(u_i)$ or with two large jobs were tested.

The computational results for one large job, $n = 100$ and $H \ge 2$ are summarized in Table 1.

The first column of this table contains the number of jobs n. Columns ILTF, LTH and ICAD contain mean value of $R_{ILTF} = C_{\max}(ILTF)/LB$, $R_{LTH} = C_{\max}(LTH)/LB$, and $R_{ICAD} = C_{\max}(ISAD)/LB$, respectively. Table 1 shows that building two schedules and choosing the best one

TABLE I
TYPE B. THE AVERAGE RELATIVE ERROR OF ALGORITHMS.

n	K	$ILTF$	LTH	$ICAD$
100	10	1.12	1.15	1.07
100	14	1.11	1.14	1.07
100	15	1.04	1.05	1.03
100	16	1.14	1.16	1.09
100	18	1.09	1.11	1.07
100	20	1.16	1.18	1.08
100	22	1.11	1.13	1.06

allows to get a schedule 6-7 percent better.

REFERENCES

[1] C. Artigues and D.Feillet, "A branch and bound method for the job-shop problem with sequence-dependent setup times", *Annals of Operations Research*, vol. 159, pp. 135—159, 2008

[2] E.Balas, J.K. Lenstra, A. Vazacopoulos, "The one-machine problem with delayed precedence constraints and its use in job shop scheduling", *Management Sci.*, vol. 41, no. 1, pp. 94 –109, 1995

[3] K.R Baker,*Introduction to Sequencing and Scheduling*, John Wiley & Son, New York, 1974.

[4] J. Carlier, "The one machine sequencing problem", *European Journal of Operational Research*, vol. 11, pp. 42—47, 1982

[5] C. Chandra, Z. Liu, J.He, T. Ruohonen, "A binary branch and bound algorithm to minimize maximum scheduling cost", *Omega*, vol. 42, pp. 9–15, 2014

[6] J. Grabowski, E. Nowicki and S. Zdrzalka, "A block approach for single-mashine scheduling with release dates and due dates", *Eur. J. Oper. Res.*, vol. 26, pp. 278–285, 1986

[7] S. Dauzere-Peres and J-B Lasserre, "A modied shifting bottleneck procedure for job-shop scheduling", *Internat. J. Production Res.*, vol. 31, no. 4, pp. 923932, 1993

[8] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey", *Ann. of Disc. Math,* vol. 5, no. 10. pp. 287–326, 1979

[9] N. Grigoreva "An 11/7- approximation algorithm for single-machine scheduling problem with release and delivery time", *Communications in Computer and Information Science*, vol. 1739, pp. 76-89, 2022

[10] N.S. Grigoreva, "Worst-Case Analysis of an Approximation Algorithm for Single Machine Scheduling Problem", *Proceedings of the 16th Conference on Computer Science and Intelligence Systems,* vol. 25, pp. 221-225. (Annals of Computer Science and Information System; v. 25), 2021.

[11] L.A. Hall and D.B. Shmoys, "Jackson's rule for single-machine scheduling: making a good heuristic better", *Mathematics of Operations Research,* vol. 17, no. 1, pp. 22—35, 1992.

[12] H. Kise, T. Ibaraki and H. Mine, "Performance analysis of six approximation algorithms for the one-machine maximum lateness scheduling problem with ready times", *Journal of the Operations Research Society of Japan*,vol, 22, pp.. 205—224, 1979.

[13] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, "Complexity of machine scheduling problems", *Ann. of Disc. Math.* vol. 1, pp. 343–362, 1977.

[14] Z. Liu, "Single machine scheduling to minimize maximum lateness subject to release dates and precedence constraints", *Computers & Operations Research,* vol. 37, pp. 1537—1543, 2010.

[15] G.B. McMahon and N. Florian, "On scheduling with ready times and due dates to minimize maximum lateness", *Operations Research.*, vol. 23, no. 3, pp. 475–482, 1975.

[16] E. Nowicki and C. Smutnicki, "An approximation algorithm for a single-machine scheduling problem with release times and delivery times", *Discrete Applied Mathematics*, vol. 48, pp. 69–79, 1994.

[17] Y. Pan and L. Shi, "Branch and bound algorithm for solving hard instances of the one-machine sequencing problem", *European Journal of Operational Research*, vol. 168, pp. 1030—1039, 2006.

[18] C.N. Potts, "Analysis of a heuristic for one machine sequencing with release dates and delivery times", *Operations Research*, vol. 28, pp. 1436—1441, 1980.

[19] Schrage L. Optimal Solutions to Resource Constrained Network Scheduling Problems (unpublished manuscript), 1971.

[20] K. Sourirajan and R. Uzsoy, "Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication", *J. Sched,* vol. 10, pp. 41-65, 2007.

[21] W. Zhang, J. Sauppe and S. Jacobson, "An Improved Branch-andBound Algorithm for the One-Machine Scheduling Problem with Delayed Precedence Constraints", *INFORMS Journal on Computing* 33, 2020. DOI: https://doi.org/10.1287/ijoc.2020.0988.