

An exact algorithm for a two-machine time-dependent scheduling problem

Weronika Skowrońska, Stanisław Gawiejnowicz

Faculty of Mathematics and Computer Science
Adam Mickiewicz University, Poznań, Poland

wersko3@st.amu.edu.pl

July 6, 2021

Outline

Introduction

Problem Formulation

Problem Properties

Exact Algorithm

Numerical Examples

Conclusions

References

Introduction I

- ▶ A number of scheduling problems exists, in which jobs processed later are more time-consuming than those started earlier.
- ▶ In the problems, jobs have variable processing times defined by functions.
- ▶ Those functions may depend on the amounts of resources allocated to the jobs, the positions of the jobs in a schedule or the job starting times.
- ▶ The latter form of job processing times is considered in **time-dependent scheduling problems**.
- ▶ Time-dependent scheduling problems appear in many applications such as scheduling maintenance procedures, multiple loan repayment problems, fire fighting problems, health care problems, recognizing aerial threats (see, e.g., Agnetis et al., 2014; Gawiejnowicz, 2020, Strusevich and Rustogi, 2017).
- ▶ Most commonly are studied time-dependent scheduling problems with job processing times defined by non-decreasing functions of the job starting times.

Introduction II

- ▶ Job processing times of this form **deteriorate** in time, i.e., a job started later has a larger processing time compared to case when it starts earlier.
- ▶ Therefore, such jobs are called **deteriorating jobs**.
- ▶ In this talk, we consider a two-machine time-dependent scheduling problem with deteriorating job processing times.
- ▶ We consider the basic properties of this problem and show how to solve it by an exact algorithm.

Problem formulation

- ▶ The problem under consideration can be formulated as follows.
- ▶ We are given a set of n independent, deteriorating jobs J_1, J_2, \dots, J_n to be scheduled on two parallel identical machines M_1 and M_2 .
- ▶ The processing time of job J_j is equal to

$$p_j(t) = 1 + b_j t, \quad (1)$$

where $t \geq 0$ is the starting time of the job and $b_j > 0$ is **deterioration rate** of J_j , $1 \leq j \leq n$.

- ▶ **For simplicity of presentation, we assume that $b_i \neq b_j$ for $1 \leq i \neq j \leq n$.**
- ▶ The aim is to find a schedule minimizing the total completion time of all jobs, $\sum_{j=1}^n C_j$, where C_j denotes the completion time of job J_j , $1 \leq j \leq n$.
- ▶ We will denote the problem as $P2|p_j = 1 + b_j t|\sum C_j$.

Problem Properties (1/2) I

- ▶ Problem $P2|p_j = 1 + b_j t| \sum C_j$ is a generalization of single machine problem $1|p_j = 1 + b_j t| \sum C_j$.
- ▶ Problem $P2|p_j = 1 + b_j t| \sum C_j$ is also a special case of problem $P2|p_j = a_j + b_j t| \sum C_j$, where $a_j \geq 0$ is the **basic processing time** of job J_j .
- ▶ The latter problem is \mathcal{NP} -hard, since in case when $a_j = 0$ it reduces to problem $P2|p_j = b_j t| \sum C_j$ which is \mathcal{NP} -hard (Kononov, 1996; Chen, 1996).

Conjecture

Problem $P2|p_j = 1 + b_j t| \sum C_j$ is \mathcal{NP} -hard.

- ▶ Based on properties of problem $1|p_j = 1 + b_j t| \sum C_j$ (Mosheiov, 1991), one can easily observe the following properties.

Problem Properties (1/2) II

Properties of problem $P2|p_j = 1 + b_j t|\sum C_j$

In an optimal schedule for problem $P2|p_j = 1 + b_j t|\sum C_j$

- ▶ the first scheduled job on each machine is the job with largest deterioration rate;
- ▶ the job with the smallest deterioration rate is scheduled neither as the second nor the last one;
- ▶ there are no three consecutively scheduled jobs J_{i-1}, J_i, J_{i+1} such that $b_i > b_{i-1}$ and $b_i < b_{i+1}$;
- ▶ the total completion time of a given schedule for a machine and the total completion time of the schedule in which all jobs excluding the first one are in reverse order are equal to each other.
- ▶ Since the next property is related to so-called **V-shaped sequences**, we briefly recall the notion.

V-shaped sequences I

- ▶ A relatively large group of scheduling problems has the property that any optimal job schedule can be divided into two parts: sequence L such that

$$p_{[1]} \geq p_{[2]} \geq \cdots \geq p_{[k]}$$

for all $J_{[i]} \in L$, followed by sequence R containing remaining jobs and such that

$$p_{[k]} \leq p_{[k+1]} \leq \cdots \leq p_{[n]}$$

for all $J_{[i]} \in R$.

- ▶ Examples of such problem include online communication (Merten & Miller, 1972), production of compounds of a product (Kanet, 1981), and production of goods that is perishable or difficult to store (Hall, 1986).

V-shaped sequences II

- ▶ Schedules of this form are optimal when there are penalties for job completion to early or to late, e.g. when we want to minimize the completion time variance or the average deviation of job completion times about a common due date (Merten & Miller, 1972; Kanet, 1981; Baker & Scudder, 1990).
- ▶ Schedules of this form are called **V-shaped schedules** and are studied for many years for fixed job processing times (Emmons, 1987; Merten & Miller, 1972) and variable job processing times (Mosheiov, 1991; Gawiejnowicz, 2020).
- ▶ V-shaped schedules on a single machine may be identified with **V-shaped sequences**.

V-shaped sequences III

- ▶ One can distinguish two types of V-shaped sequences: **proper** and **improper**.
- ▶ Every proper V-shaped sequence is composed of two branches, one containing jobs in descending order, the second one in ascending order.
- ▶ For a sequence of length n , there exists $2^{n-1} - 2$ proper V-shaped sequences.
- ▶ Every improper V-shaped sequence does not contain either the right (descending) or the left (ascending) part of the sequence.
- ▶ For a sequence of length n , there exists 2 improper V-shaped sequences.
- ▶ The largest element of a V-shaped sequence appears either on the left or on the right end of the sequence.

Problem Properties (2/2)

- ▶ Now, based on the properties, we can formulate the next result.

Theorem

An optimal schedule for problem $P2|p_j = 1 + b_j t|\sum C_j$ is V-shaped on each machine.

- ▶ The result may be used in a recursive algorithm generating all V-shaped schedules for the problem.
- ▶ Though it is an exponential algorithm, it may be useful in studying the structure of optimal schedules.

Exact algorithm

Algorithm for one machine

- ▶ The main idea of an exact algorithm for the problem is as follows.
- ▶ First, we find a partition of jobs between two machines.
- ▶ Next, for the partition we schedule the jobs independently on each machine.
- ▶ We repeat the two steps for all possible partitions of jobs, and looking for the lowest value of criterion $\sum C_j$ we find an optimal schedule.

For generating V-shaped sequences for one machine scheduling problem we have proposed (Skowrońska, Gawiejnowicz, 2020) a recursive algorithm working as follows:

- ▶ When having a sorted sequences of length n , we take the smallest element to be in the middle.
- ▶ Then, we recursively assign following elements to either left or right part of an array to form a V-shaped sequence.

Exact algorithm

Pseudocode: Recursive algorithm for one machine problem

Algorithm 1 RA_ONE_M(N)

Require: N - length of the sequence

$K = 2$ - next index

$I = [1]$ - Initial array

$RESULTS = []$ Array for storing results

function GEN_V_SHAPE($INP_ARRAY, N, RESULTS, K$)

$LEFT = CONCATENATE(K, INP_ARRAY)$

$RIGHT = CONCATENATE(INP_ARRAY, K)$

if $K = N$ **then**

$RESULTS = CONCATENATE(LEFT, RIGHT)$

return

end if

 GEN_V_SHAPE($LEFT, N, RESULTS, K+1$)

 GEN_V_SHAPE($RIGHT, N, RESULTS, K+1$)

return $RESULTS$

end function

Exact algorithm

Algorithm for two machines

For the two machine problem with criterion function, we use the previously proposed algorithm with few modifications:

- ▶ At the beginning, we generate all possible divisions of the set of jobs for two machines.
- ▶ We calculate and store the value of the criterion function.

Exact algorithm

Pseudocode: Calculating C_j

Algorithm 2 COMPUTE_C_J

```
function COMPUTE_C_J( $b_j, t$ )  
    return  $1 + (1 + b_j) * t$   
end function
```

Exact algorithm

Pseudocode: Recursive algorithm for two machine problem

Algorithm 3 TWO_MACHINE_ALGORITHM

Require: N - length of the sequence

$K = 2$ - next index

$I = [1]$ - Initial array

$OPT = [[+INF], [0]]$ Array for storing optimum sequence and the criterion value.

f - job processing time function

Exact algorithm

Pseudocode: Recursive Algorithm for two machine problem

```
function GEN_V_SHAPE(INP_ARRAY, CURR_ARRAY, N, OPT, f, K)
  new_val = INP_ARRAY[K-1]
  LEFT = CONCATENATE(CURR_ARRAY, new_val)
  RIGHT = CONCATENATE(new_val, CURR_ARRAY)
  if  $K = N$  then
    RESULTS = CONCATENATE(LEFT, RIGHT)
    SUM_CJ = 0
    for I in RESULTS do
      SUM_CJ = 0
      for J in I do
        SUM_CJ = SUM_CJ + f(I, SUM_CJ)
      end for
      if SUM_CJ < OPT[0][0] then
        OPT[0][0] = SUM_CJ
        OPT[0][1] = I
      end if
    end for
  return
end if
```

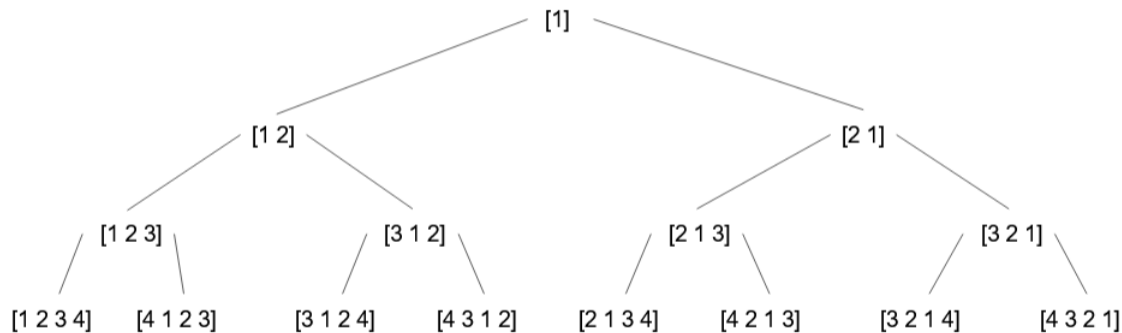
Exact algorithm

Pseudocode: Recursive Algorithm for two machine problem

```
GEN_V_SHAPE(INP_ARRAY, LEFT, N, OPT, f, K+1)
GEN_V_SHAPE(INP_ARRAY, RIGHT, N, OPT, f, K+1)
return RESULTS, OPT
end function
```

Numerical examples

Example of one machine problem



Numerical examples

Example of two machine problem

$$\begin{aligned}B_1 &= 8 \\B_2 &= 4 \\B_3 &= 3 \\B_4 &= 7 \\B_5 &= 5\end{aligned}$$

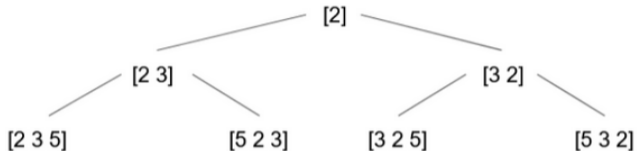
[[1 3 4 5] [2]]
[[1 2 4 5] [3]]
[[1 2 3 5] [4]]
[[1 2 3 4] [5]]
[[3 4 5] [1 2]]
[[2 4 5] [1 3]]
[[2 3 5] [1 4]]
[[2 3 4] [1 5]]
[[1 4 5] [2 3]]
[[1 3 5] [2 4]]
[[1 3 4] [2 5]]
[[1 2 5] [3 4]]
[[1 2 4] [3 5]]
[[1 2 3] [4 5]]

Numerical examples

Example of two machine problem

$B_1 = 8$
 $B_2 = 4$
 $B_3 = 3$
 $B_4 = 7$
 $B_5 = 5$

[[1 3 4 5] [2]]
[[1 2 4 5] [3]]
[[1 2 3 5] [4]]
[[1 2 3 4] [5]]
[[3 4 5] [1 2]]
[[2 4 5] [1 3]]
[[2 3 5] [1 4]]
[[2 3 4] [1 5]]
[[1 4 5] [2 3]]
[[1 3 5] [2 4]]
[[1 3 4] [2 5]]
[[1 2 5] [3 4]]
[[1 2 4] [3 5]]
[[1 2 3] [4 5]]

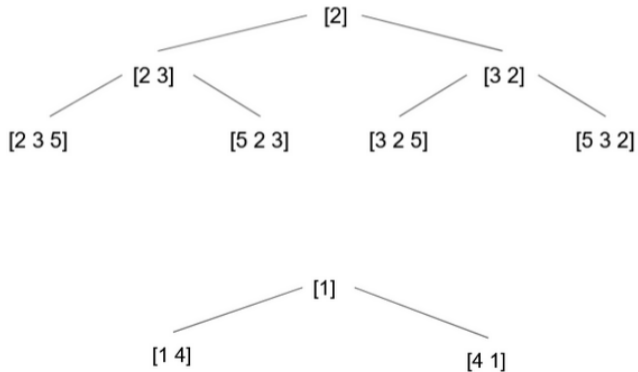


Numerical examples

Example of two machine problem

$B_1 = 8$
 $B_2 = 4$
 $B_3 = 3$
 $B_4 = 7$
 $B_5 = 5$

[[1 3 4 5] [2]]
[[1 2 4 5] [3]]
[[1 2 3 5] [4]]
[[1 2 3 4] [5]]
[[3 4 5] [1 2]]
[[2 4 5] [1 3]]
[[2 3 5] [1 4]]
[[2 3 4] [1 5]]
[[1 4 5] [2 3]]
[[1 3 5] [2 4]]
[[1 3 4] [2 5]]
[[1 2 5] [3 4]]
[[1 2 4] [3 5]]
[[1 2 3] [4 5]]

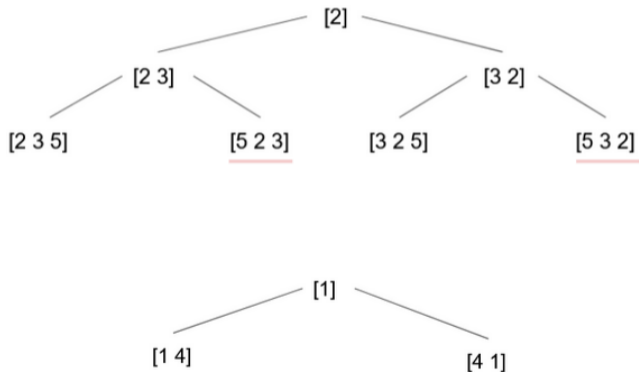


Numerical examples

Example of two machine problem

$B_1 = 8$
 $B_2 = 4$
 $B_3 = 3$
 $B_4 = 7$
 $B_5 = 5$

[[1 3 4 5] [2]]
[[1 2 4 5] [3]]
[[1 2 3 5] [4]]
[[1 2 3 4] [5]]
[[3 4 5] [1 2]]
[[2 4 5] [1 3]]
[[2 3 5] [1 4]]
[[2 3 4] [1 5]]
[[1 4 5] [2 3]]
[[1 3 5] [2 4]]
[[1 3 4] [2 5]]
[[1 2 5] [3 4]]
[[1 2 4] [3 5]]
[[1 2 3] [4 5]]

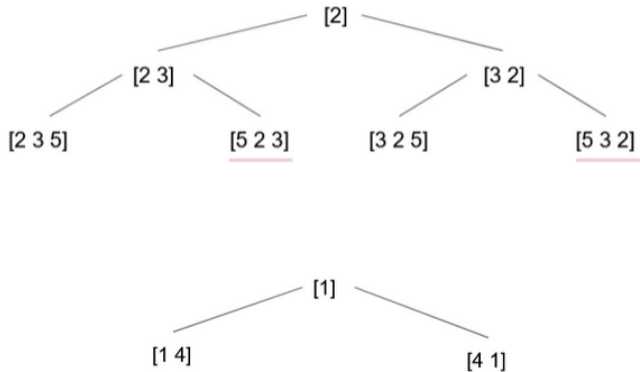


Numerical examples

Example of two machine problem

$B_1 = 8$
 $B_2 = 4$
 $B_3 = 3$
 $B_4 = 7$
 $B_5 = 5$

[[1 3 4 5] [2]]
[[1 2 4 5] [3]]
[[1 2 3 5] [4]]
[[1 2 3 4] [5]]
[[3 4 5] [1 2]]
[[2 4 5] [1 3]]
[[2 3 5] [1 4]]
[[2 3 4] [1 5]]
[[1 4 5] [2 3]]
[[1 3 5] [2 4]]
[[1 3 4] [2 5]]
[[1 2 5] [3 4]]
[[1 2 4] [3 5]]
[[1 2 3] [4 5]]









Conclusions





In this talk:

- ▶ We considered a two-machine time-dependent scheduling problem.
- ▶ We formulated the basic properties of the problem.
- ▶ We presented an exact algorithm for the problem, based on an algorithm creating all V-shaped sequences for one machine case.
- ▶ We reported the results of numerical experiments with implementations of discussed algorithms in Python 3.8.
- ▶ Future research may concern establishing the time complexity of the problem, finding a good lower bound on the optimality criterion value or the construction of a branch-and-bound algorithm.



References

-  A. Agnetis, J-C. Billaut, S. Gawiejnowicz, D. Pacciarelli, A. Soukhal, Multiagent Scheduling: Models and Algorithms, Springer, 2014.
-  K.E. Baker, G.D. Scudder, Sequencing with earliness and tardiness penalties: a review, Operations Research, 38 (1990), 22-38.
-  Z.L. Chen, Parallel machine scheduling with time dependent processing times, Discrete Applied Mathematics, 70 (1996), 81-93; 75 (1997), 103.
-  H. Emmons, Scheduling to a common due date on parallel common processors, Naval Research Logistics Quarterly, 34 (1987), 803-810.
-  S. Gawiejnowicz, A review of four decades of time-dependent scheduling: main results, new topics, and open problems, Journal of Scheduling, 23 (2020), 3-47.
-  S. Gawiejnowicz, Models and Algorithms of Time-Dependent Scheduling, Springer, 2020.

References

-  N. Hall, Single and multi-processor models for minimizing completion time variance, *Naval Research Logistics Quarterly*, 33 (1986), 49–54.
-  J.J. Kanet, Minimizing the average deviation of job completion times about a common due date, *Naval Research Logistics Quarterly*, 28 (1981), 643-651.
-  A. Kononov, Scheduling problems with linear increasing processing times, in: *Operations Research 1996*, Springer, 1997, 208–212,
-  A.G. Merten, M.E. Muller, Variance minimization in single machine sequencing problems, *Management Science*, 18 (1972), 518-528.

References

-  G. Mosheiov, V-Shaped policies for scheduling deteriorating jobs, *Operations Research*, no. 6, 39 (1991), 979–991.
-  W. Skowronska, S. Gawiejnowicz, Algorithms generating V-shaped sequences and their applications, *ECCO 2021*.

Thank you for your attention!