14th Annual International Conference on Industrial Engineering and Operations Management Dubai United Arab Emirates (UAE), February 12-14, 2024

Publisher: IEOM Society International, USA
Published: February 12, 2024
DOI: 10.46254/AN14.20240064

A Branch and Bound Algorithm to Optimize Multi-Objective Competitive Multi-Agent Scheduling with Simultaneous Log-linear Position-based Learning and Time-based Deterioration Effects

Tugba Danaci
Fatma Senses VS Business Administration Department
Member, Public University-Industry Collaboration Committee
Kirikkale University
Kirikkale, TURKIYE
DEDA Technology Investment Ltd
Ankara, TURKIYE

Abstract

Encompassing a subset of multi-objective scheduling challenges, the multi-agent scheduling problem involves various agents, each entrusted with a unique set of tasks while striving to optimize their individual goals. Recent inquiries in this field have predominantly spotlighted variable processing times, employing methods like the ϵ-constraint approach to optimize one agent's function without compromising the other's limit. Our study takes an innovative approach, delving into a two-agent single-machine scheduling problem influenced by concurrent learning and deterioration effects. The primary aim is to minimize the overall weighted completion time for both agents, preventing any job delays for the second agent. To address this, our research amalgamates the ϵ-constraint and linear combination approaches, presenting a unique proposition in the current research landscape. We introduce a two-stage methodology: a heuristic method for near-optimal solutions followed by a branch-and-bound algorithm, integrating specialized dominance rules to achieve optimal solutions.

Keywords

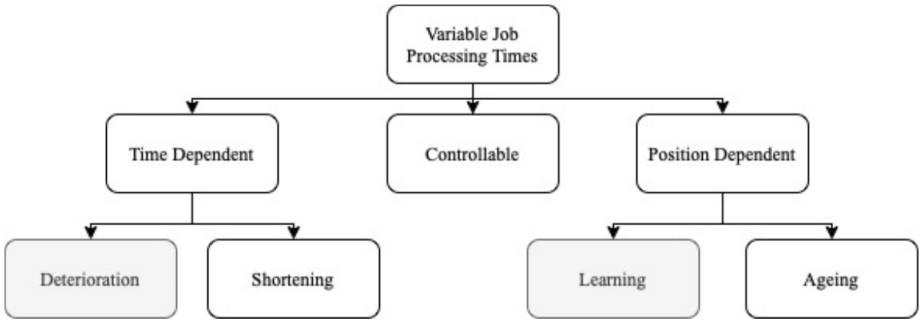
Multi-agent scheduling, Variable processing times, Learning effect, Deterioration effect, Branch-and-bound algorithm

1. Introduction

In classical scheduling problems, job processing times are assumed to remain constant and stable. However, in practical situations, these processing times can fluctuate due to factors such as job repetitions or variations in resource allocation. The literature categorizes variations in processing times into three primary groups: time-dependent processing times, position-dependent processing times, and controllable processing times (Agnetis et al., 2014).

Time-dependent processing times are determined by when a job begins processing, while position-dependent processing times are influenced by the job's starting position or row. Additionally, processing times can sometimes be altered by adjusting the number of allocated resources, leading to jobs with controllable processing times. Time-dependent processing times can further be classified into two subgroups: non-decreasing, where processing times increase as jobs are delayed (referred to as jobs under deterioration effect), and non-increasing, where processing times decrease with delays (referred to as jobs under shortening effect). Position-dependent processing times are categorized into non-decreasing (jobs under aging effect) and non-increasing (jobs under learning effect) groups based on worker experience. Furthermore, certain jobs, particularly those in project management, allow for variations in

processing times based on resource usage. For a mor detailed breakdown of job categorization based on varying



processing times, please refer to Figure 1.

Figure 1. Groups of Jobs with Variable Processing Times (Agnetis et al., 2014)

1.1 Objectives

In this study, our objective is to minimize the total weighted completion time in a single-machine scheduling problem involving two competing agents, considering jobs affected by both learning and deterioration effects simultaneously, while ensuring that the second agent does not encounter tardy jobs. The processing times of the jobs are influenced by two concurrent factors: log-linear position-based learning and linear time-based deterioration. The deterioration effect causes the processing time of a job to increase as it is scheduled later in the sequence, while the learning effect results in a reduction of job processing times.

2. Literature Review

Multi-agent scheduling problem is a subset of multi-objective scheduling problems where each agent has a set of jobs, and its objective is to optimize its own objective function. The literature on multi-agent scheduling problems can be divided into two main groups based on the characteristics of the processing time of the jobs: problems with non-variable processing times and problems with variable processing times. The paper of Agnetis and others (Agnetis et al., 2000; Agnetis et al., 2001) and Baker & Smith (Baker et al., 2003) can be considered as pioneer studies for multi-agent scheduling problems. In reviewing the literature studies, we examined papers that study the same objective function as ours - minimizing the total weighted completion time.

The first group of the literature review includes the studies on jobs with non-variable processing times. Soltani and others studied to minimize the total weighted completion time for the first agent while minimizing the maximum lateness for the second agent (Soltani et al., 2010). They attempted to minimize the total weighted completion time for the first agent with a threshold for the maximum latenss value of the second agent by proposing a two-step methodology: simulated annealing and branch-and-bound algorithms. The same problem was also studied by Yin and others (Yin et al., 2015). They proposed an algorithm consisting of the honeybee algorithm and the branch and bound

algorithm. Lee and Wang (2014) studied the problem with three agents. They tried to minimize the total weighted completion time for the first agent with a threshold for the total completion time value of the second agent while the maintenance process of the third agent should have been finished in a predefined time interval. They also proposed a two-stage methodology. In the first stage, to obtain an initial solution, a local search and genetic algorithm were used. In the second stage, they used branch and bound algorithms. Another paper belongs to Choi and Chung (2014). They tried to minimize not only the total weighted completion time but also the weighted number of tardy jobs of the first agent subject to the restriction of the weighted number of just-in-time jobs of the second agent. They studied the complexity level of the problem.

The second group of the literature review includes the studies on jobs with variable processing times. The papers closest to our problem in terms of objective functions studied the minimization of the total weighted completion time for the first agent under the restriction that no tardy job is allowed for the second agent (Lee at al., 2010; Cheng et al., 2011; Wu et al., 2013; Wu et al., 2014). The jobs of both agents are under the effect of linear deterioration in the study by Lee and others (Lee at al., 2010) and under the effect of log-linear learning in the study by Wu and others (Wu et al., 2013). In the study by Cheng and others (Cheng et al., 2011), the first agent's jobs are under the effect of log-linear learning and the second agent's jobs are under the effect of log-linear aging, while in the study by Wu and others (Wu et al., 2014), the first agent's jobs are under the effect of past-dependent learning and the second agent's jobs are under the effect of past-dependent aging. Danaci and Toksari, on the other hand, studied with the jobs that are under the simultaneous effect learning and deterioration (Danaci & Toksari, 2021).

Few articles have studied the total weighted completion time for both agents (Lee et al., 2009; Nong et al., 2011; Wu, 2014). In their study, Lee et al. studied jobs with non-variable processing times (Lee et al., 2009). They reduced the problem to the multi-objective shortest path problem and developed a solution approach with polynomial time. In their study, Nong et al. developed a global objective function by summing the objective functions of the two agents. They showed the problem is NP-hard and developed two different approximation algorithms. In the study belonging to Wu, jobs under the effect of past-sequence learning are studied with the constraint that the makespan of the second agent must not exceed a certain upper bound (Wu, 2014). Zhang et al. studied scheduling with three agents on a single machine in which the criteria of the three agents are to minimize the total weighted completion time, the weighted number of tardy jobs, and the total weighted late work, respectively (Zhang et al., 2020). Since the problem was NP-hard, they studied the problem under the assumption that the jobs of the first agent have inversely agreeable processing times and weights. The smaller the processing time of a job was, the greater its weight was. They presented a pseudo-polynomial-time algorithm to find the Pareto frontier.

3. Methods

In this paper, we endeavor to minimize the total weighted completion time for both agents, taking into account jobs affected by both learning and deterioration effects simultaneously, while ensuring that the second agent does not encounter tardy jobs. To achieve this, we formulate a global objective function through weighted summation of each agent's objective functions. Our solution approach comprises two stages. In the initial stage, we develop a heuristic to generate an initial solution, with the objective function value serving as input for the second stage. In the second stage, we employ a branch-and-bound algorithm enriched with multiple dominance rules and a lower bound to identify the optimal solution. Finally, we conduct computational experiments to evaluate the performance of the proposed algorithms.

4. Problem Description

The problem we are addressing is a single-machine scheduling problem involving two competing agents, denoted as Agent A and Agent B. There is a total of + jobs, where "" jobs belong to Agent A, and "" jobs belong to Agent B. All jobs become available for processing at time "t=0," and no interruptions are allowed during the scheduling process. The processing times of the jobs are influenced by two concurrent factors: log-linear position-based learning and linear time-based deterioration. As a result of the deterioration effect, the processing time of a job increases as it is scheduled later in the sequence. Conversely, the learning effect leads to a reduction in the processing time of jobs.

The notations and variables used throughout this paper are as follows:

J_A The job set for agent A

J_B	The job set for agent B
$J = J_A \cup J_B$	The set for the sum of jobs
n_A	The number of jobs for agent A
n_B	The number of jobs for agent B
$n = n_A + n_B$	The total number of jobs
p_j	Normal processing time for job j
d_j	Due date for job j
w_j^A	Weight for job j of agent A
w_j^B	Weight for job j of agent B
w_A	Weight for agent A
w_B	Weight for agent B
β	Deterioration coefficient ($\beta \geq 0$)
α	Learning coefficient ($\alpha \leq 0$)
r	Position in the schedule
t	Time when job j is started to be processed
$p_j(r,t) = \left(p_j + \beta t\right)r^{\alpha}$	The real processing time of job j is started to be processed in position r and time t
S	The present schedule
$C_j^A(S)$	The completion time of the job j of agent A in the schedule S
$C_j^B(S)$	The completion time of the job j of agent B in the schedule S
$U_j(S)$	It's 1 if the job j is tardy in the schedule S, it's 0 if the job j is not tardy
$OFV(S)$	Objective Function Value in schedule S
$k = k_A + k_B$	The number of assigned jobs to the node in branch and bound algorithm
k_A	The number of unassigned jobs belonging to agent A to the node in BBA
k_B	The number of unassigned jobs belonging to agent B to the node in BBA
$\hat{C}_{[r,t]}$	The completion time of the job when started to be processed in position r and time t
UB	Upper bound value

The primary objective function explored in this paper aims to minimize the weighted combination of the total weighted completion time of both agents while imposing the constraint that no jobs for Agent B are allowed to become tardy. The specific problem under investigation can be formulated as follows:

$$1/CO; p_j(r,t) = \left(p_j + \beta t\right)r^{\alpha}; \sum_{j=1}^{n_B} U_j^B(S) = 0 / w_A \sum_{j=1}^{n_A} w_j^A C_j^A(S) + w_B \sum_{j=1}^{n_B} w_j^B C_j^B(S) \quad \text{where } w_j^A + w_j^B = 1$$

It is worth noting that the problem at hand is NP-hard. In order to address this challenging problem, a methodology centered around the branch and bound algorithm (BBA) has been devised. In the first phase of this methodology, a heuristic approach has been formulated to generate an initial solution. The objective function value derived from this initial solution is subsequently utilized as the upper bound for the branch and bound algorithm.

5. Heuristics for Initial Solution

As previously mentioned, the objective function value of the initial solution plays a pivotal role as the first upper bound for the branch and bound algorithm. Consequently, the effectiveness and efficiency of the branch-and-bound algorithm are closely tied to the quality of this initial solution. In this section, we will delve into the development and implementation of heuristics aimed at generating a high-quality initial solution. The quality of this initial solution not only impacts the overall efficiency of the algorithm but also significantly influences the algorithm's ability to converge towards an optimal or near-optimal solution.

To obtain the initial solution, for each position in the schedule, the unassigned jobs are sorted by the WSPT rule with respect to $pd_j(r,t) = (p_j + \beta t)r^{\alpha}/(w_j^A w_A)$ if $J_j \in J_A \cap J$ and $pd_j(r,t) = (p_j + \beta t)r^{\alpha}/(w_j^B w_B)$ if $J_j \in J_B \cap J$. The unassigned jobs of Agent B are sorted according to the EDD rule. The job with the weighted smallest processing time is selected. If none of the unassigned jobs of agent B gets tardy; the selected job is assigned to the position. Otherwise, the job belonging to Agent B with the smallest deadline is selected and assigned to the position. The pseudocode of the heuristics developed to obtain the initial solution is shown in Algorithm 1 and the flowchart in Figure 2.

Algorithm 1. Heuristics for Initial Solution

Input: $J = \{J_1, J_2, \dots, J_n\}$: set of all jobs
$S = \{\dots\}$: partial schedule
$t = 0$: completion time
For $(r \leftarrow 1 to\ n)$ **do**

1. Set adjusted processing times for each job with respect to the following: If $J_j \in J_A \cap J$ then $pd_j(r,t) = (p_j + \beta t)r^{\alpha}/(w_j^A w_A)$; If $J_j \in J_B \cap J$ then $pd_j(r,t) = (p_j + \beta t)r^{\alpha}/(w_j^B w_B)$
2. Select the job with min{pd(u)} value for $J_u \in J$.
3. Calculate the real processing time $p_j(r,t) = \left(p_j + \beta t\right)r^{\alpha}$ and completion time $C^{\wedge}_{[r,t]} = p_j(r,t) + t$ of J_u as assigned to position *r* in partial schedule S.
4. Check whether adding job J_u to the partial schedule S would result in tardiness for any unscheduled jobs belonging to Agent B by using Algorithm 2.
5. If $number - of - tardy - jobs = 0$, assign the job J_u to r^{th} position in the partial schedule S and remove it from the set of unscheduled jobs J. Proceed to step 9.
6. Else, select the job with the smallest due date from the set *tardy-control* $= \{j \mid j \in J_B \cap J, j \notin S\}$ and remove it from the set. Then, reapply Algorithm 2 to check the impact of this new assignment.
7. If the number of tardy jobs is reduced to zero after assigning the job with the smallest due date, assign that job to r^{th} position in the partial schedule "S" and remove it from the set of unscheduled jobs "J." Proceed to step 9.
8. Else, the solution is INFEASIBLE.
9. Calculate the completion time of the partial schedule and update *t.*
10. $r = r + 1$

End
Output $\boldsymbol{S} = \{\boldsymbol{J}_{[1]}\boldsymbol{J}_{[2]}, \dots, \boldsymbol{J}_{[n]}\}$
Output $UB = OFV(S) = w_A \sum_{j=1}^{n_A} w_j^A C_j^A(S) + w_B \sum_{j=1}^{n_B} w_j^B C_j^B(S)$

Algorithm 2. Tardy control algorithm

Input: $J = \{J_1, J_2, \dots, J_n\}$
$S = \{\dots, J[r-2], J[r-1]\}$
tardy-control $= \{j \mid j \in J_B \cap J, j \notin S\}$
number-of-tardy-jobs $= 0$

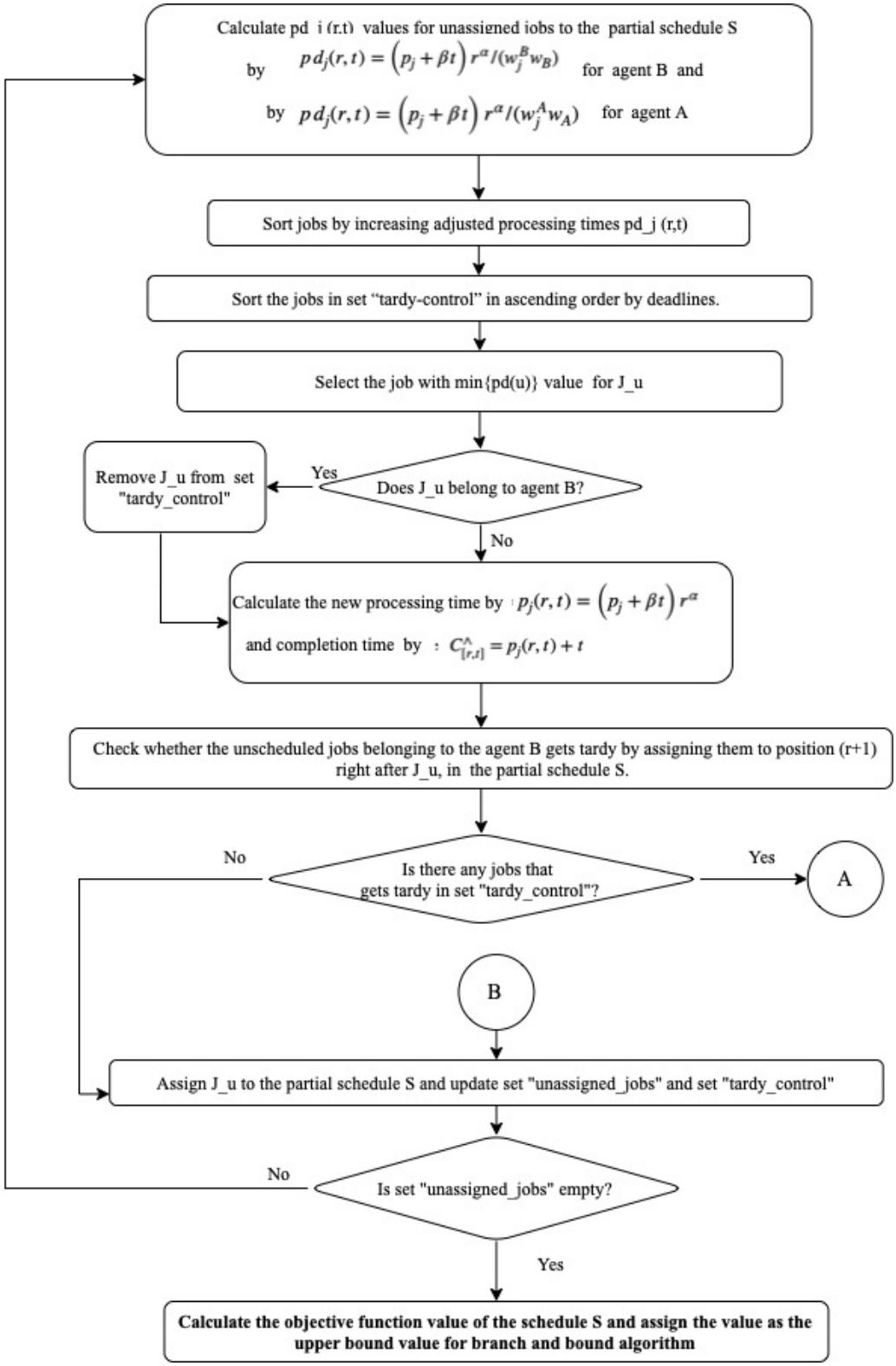
1. Sort the jobs in set "tardy-control" in ascending order based on their deadlines.
2. Schedule the jobs from the "tardy-control" set to the partial schedule S right after the job J_u

For $j \in tardy - control \cap S$ **do:**

Calculate the real processing time $p_j(r,t) = \left(p_j + \beta t\right) r^{\alpha}$ and completion time $C^{\wedge}_{[r,t]} = p_j(r,t) + t$ of job j

if $C_j(S) > d_j$ **:**
$number - of - tardy - jobs = +1$
Break

End
Output number_of_tardy_jobs



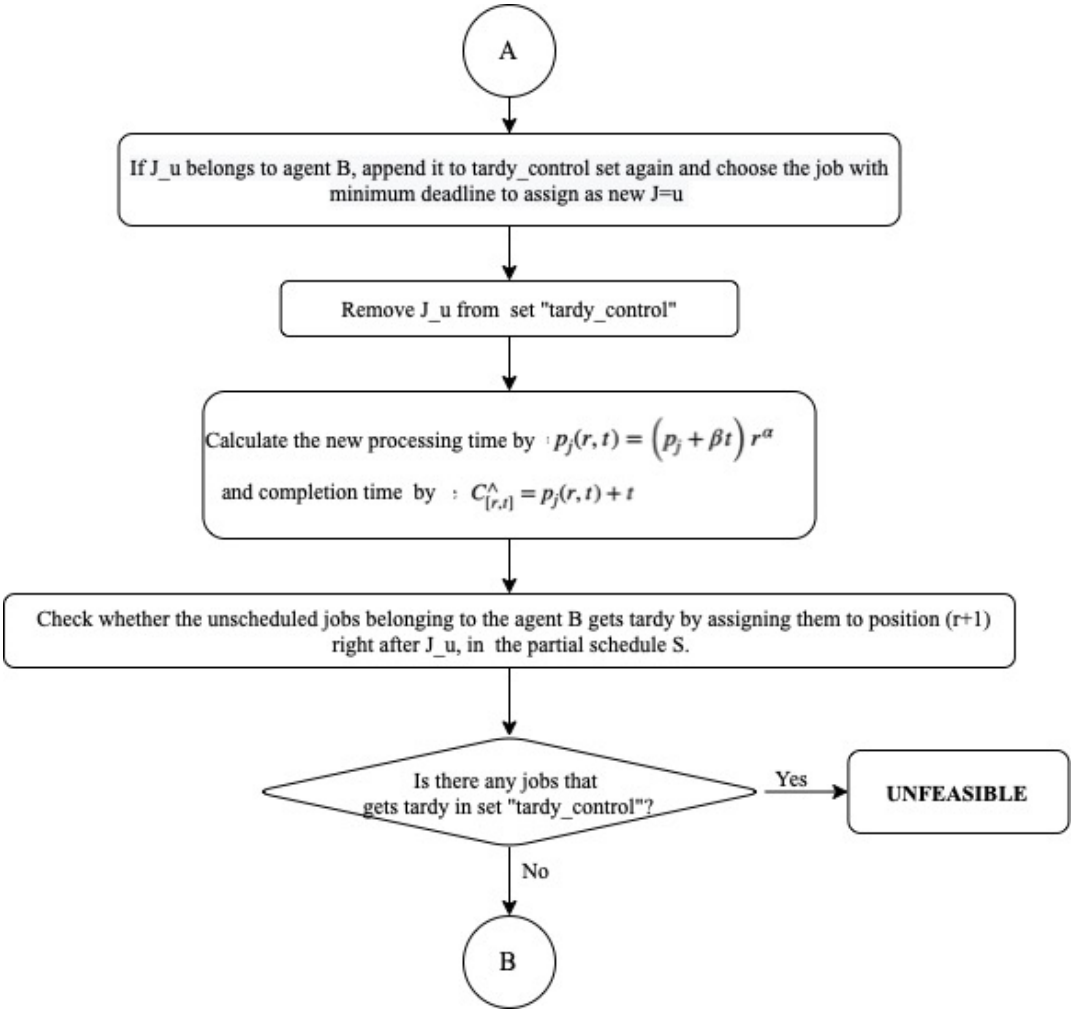


Figure 2. Flowchart of the Proposed Initial Solution

6. The Branch and Bound Algorithm

In the branch-and-bound algorithm designed to address this scheduling problem known for its NP-hard complexity, multiple pruning rules characterized by dominance properties have been incorporated to streamline the branching procedure. These pruning rules serve to diminish the search space, facilitating a more efficient exploration of potential solution candidates.

6.1 Pruning Rules

Here is an outline of the implemented pruning rules utilized within the algorithm:

- ***Pruning by Boundaries:*** This rule mandates that the calculated lower bound value of the partial schedule in a node must be less than or equal to the best-known upper bound value. If this condition is not met, the node undergoes pruning. The initial solution's objective function value serves as the initial upper bound value. Detailed information on how to compute the lower bound for each node can be found in Section 6.2 and Algorithm 4.

- ***Pruning by Feasibility:*** The problem's objective is to minimize the total weighted completion time for both agents while ensuring that no tardy job occurs for the second agent. If any unassigned job of the second agent

becomes tardy due to the partial schedule in a node, the solution is deemed infeasible, leading to node pruning. The tardiness verification for unassigned jobs is elaborated in Algorithm 2.

- ***Pruning by Optimality:*** Once all jobs belonging to the second agent have been assigned to a node, the optimal schedule is achieved using the Weighted Shortest Processing Time (WSPT) rule. To allocate the remaining unassigned jobs to the partial schedule, the jobs are sorted based on $\boldsymbol{pd_j(r,t) = (p_j + \beta t)r^{\alpha}/w_j^A}$ values. This approach significantly reduces the number of nodes generated compared to conventional branching techniques, thereby enhancing efficiency.

- ***Pruning by Dominance:*** Non-dominant nodes are pruned to further restrict the search space. The properties of dominance are elucidated in greater detail in Section 6.3 and Algorithm 5.

These pruning rules serve as guiding principles for the branch-and-bound algorithm, directing it towards promising solution paths while eliminating less promising branches early in the search process. This contributes to enhanced algorithm efficiency and enables it to handle the inherent complexity of NP-hard problems, as demonstrated in the paper.

The pseudo code of the branch and bound algorithm is given in Algorithm 3. Flowchart is given in Figure 3.

Algorithm 3. Branch and Bound Algorithm

Input: $J = \{J_1, J_2, \dots, J_n\}$
node-list = {Partial schedules obtained by branching nodes in branch and bound algorithm}
UB = objective function vale of the schedule obtained by using Algorithm 1
For $(r \leftarrow 1 to\ n)$ **do**
For node in node-list do:
1. For each partial schedule, create the set of "unassigned-jobs"
2. For the unassigned job belongs to Agent B; create "unassigned-B-agent-jobs" set.
3. Sort jobs in "unassigned-B-agent-jobs" set by EDD rule

if *unassigned-jobs* $\subset J_A$ **:**
1. Sort jobs by $pd_j(r,t) = (p_j + \beta t) r^{\alpha} / w_j$
2. Obtain schedule S that contain all the jobs by adding the jobs in "unassigned-jobs" set in the same order. Calculate OFV (S).

if OFV(S) <= UB:
1. Equal UB to OFV(S)
2. Add schedule S to "feasible-solutions" set.

Else:
For each job in "unassigned-jobs" do:
1. Add the job to the partial schedule S in the node.
2. Calculate lower bound value of S with Algorithm 4.

If LB <= UB
Check if any of the unassigned jobs belongs to agent B gets tardy by using Algorithm 2.
If number_of_tardy_jobs = 0:
Apply Algorithm 5 to check if the partial schedule S is dominant.
If S is dominant in node-list:
Add S to the node-list**.**
If s(S/J) = 0:
Equal LB to UB
Add schedule S to "feasible-solutions" set.
Else Prune the node
Else Prune the node
Else Prune the node

End
Output node-list
End
End
Output feasible-solutions
1.Sort schedules in "feasible-solutions" set in ascending order of objective function values.
2.Add schedule or schedules with the smallest objective function value to the "optimal-solution" set.
End
Output optimal-solution

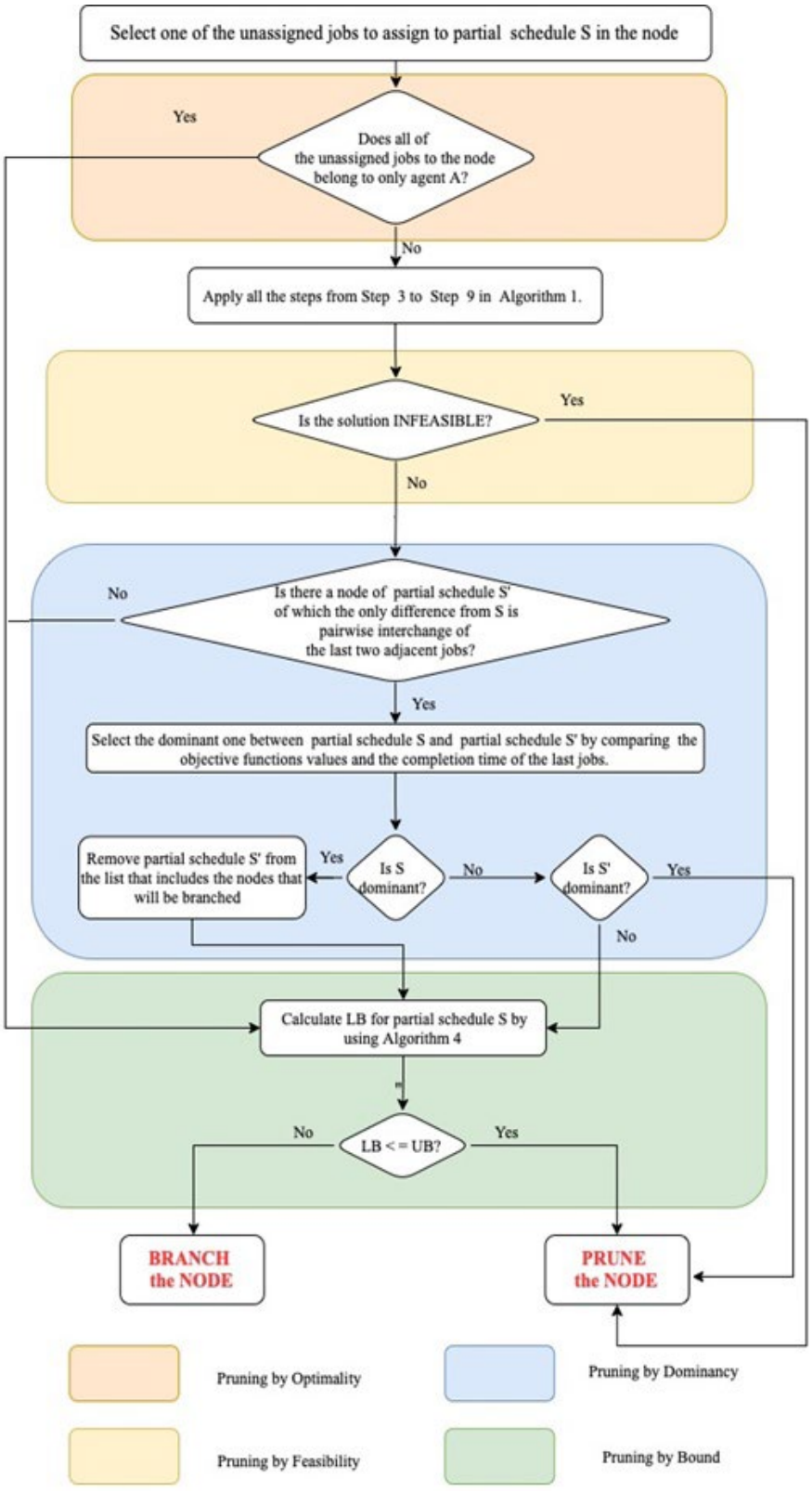


Figure 3. Pruning Rules

6.2 Calculation of Lower Bound

The efficiency of the branch-and-bound algorithm relies significantly on the accurate calculation of a lower bound for the partial sequence. Let's consider a partial schedule denoted as "S," where the order of the first "k" jobs is already determined, leaving "(n - k)" jobs unassigned to the partial schedule within the current node. The completion time for the partial schedule S is represented by $\boldsymbol{C_{[k]}}$.

If there were not a tardiness constraint on Agent B's jobs; the optimal schedule would be obtained by the WSPT method with the real processing times of the jobs are calculated for each position because of simultaneous effect of learning and deterioration. In other words, the objective function of the problem studied in this paper is always larger than or equivalent to the problem with no tardiness constraint.

The pseudo code for calculating the lower bound value is given in Algorithm 4.

Algorithm 4. Calculation of Lower Bound

$C_{[k]}$: the completion time of partial schedule S

For $(r \leftarrow 1 to n-k)$ **do:**

1. Calculate adjusted processing times for each unassigned job with respect to the following: If $J_j \in J_A \cap J$ then $pd_j(r,t) = \left(p_j + \beta C_{[k+r-1]}\right)(r+k)^{\alpha}/(w_j^A w_A)$; If $J_j \in J_B \cap J$ then $pd_j(r,t) = \left(p_j + \beta C_{[k+r-1]}\right)(r+k)^{\alpha}/(w_j^B w_B)$
2. Assign the job with minimum value $pd_j(r,t)$ to the position $(k+r)$ of partial schedule S and remove the job from "unassigned -jobs" set.
3. Calculate the completion time for the job with $C_{[k+r]} = \left(p_j + \beta C_{[k+r-1]}\right)(r+k)^{\alpha} + C_{[k+r-1]}$
4. r = r +1

Calculate lower bound:

$$LB = w_A \sum_{j=1}^{k} w_{[k]}^A C_{[k]}^A + w_B \sum_{j=1}^{k} w_{[k]}^B C_{[k]}^B + w_A \sum_{r=1}^{n-k-k_B} w_{[r]}^A C_{[r]}^A + w_B \sum_{r=1}^{k_B} w_{[r]}^B C_{[r]}^B$$

End

Output $LB = w_A \sum_{j=1}^{k} w_{[k]}^A C_{[k]}^A + w_B \sum_{j=1}^{k} w_{[k]}^B C_{[k]}^B + w_A \sum_{r=1}^{n-k-k_B} w_{[r]}^A C_{[r]}^A + w_B \sum_{r=1}^{k_B} w_{[r]}^B C_{[r]}^B$

6.3 Dominance Rules

In the context of dominance rules within the branch-and-bound algorithm, let's consider two partial schedules, denoted as S and S', with the only difference being the exchange of two adjacent jobs, i and j, in pairs. In partial schedule S, job i is processed at the $\boldsymbol{r^{th}}$ position while job j is processed at the $\boldsymbol{(r+1)^{th}}$ position. We denote "t" as the completion time of the job scheduled at the $\boldsymbol{(r-1)^{th}}$ position.

According to the dominance rules, new job assignments continue on the node with the dominant partial schedule and the node with the non-dominant schedule is pruned. Since the jobs are simultaneously under the effects of learning

and deterioration; 2 parameters determine the dominance: the objective function value of partial schedule and the completion time of the last job scheduled in the partial schedule.

Partial schedule S is considered dominant over S ’if any of the conditions listed below are met:

- $\boldsymbol{OFV(S) < OFV(S')}$ and $\boldsymbol{C_j(S) < C_i(S')}$
- $\boldsymbol{OFV(S) < OFV(S')}$ and $\boldsymbol{C_j(S) = C_i(S')}$
- $\boldsymbol{OFV(S) = OFV(S')}$ and $\boldsymbol{C_j(S) < C_i(S')}$

In other combinations of conditions where these criteria are not satisfied, it may be challenging to definitively determine which partial schedule is dominant. However, when one schedule is dominant over the other, the branch-and-bound algorithm prioritizes further exploration and assignment of new jobs within the dominant schedule, ultimately contributing to the search for an optimal solution.

Algorithm 5. Dominance Rules

Input: $S = \{\dots, J_{[r-2]}, J_{[r-1]}, J_{i,[r]}\}$
$S' = \{\dots, J_{[r-2]}, J_{[r-1]}, J_{j,[r]}, J_{j,[r+1]}\}$
node-list = {Partial schedules obtained by branching nodes in branch and bound algorithm}

For J_j **do:**

1. Add job J_j to partial schedule S, position (r+1)
2. Get $S = \{\dots, J_{[r-2]}, J_{[r-1]}, J_{i,[r]}, J_{j,[r+1]}\}$

If S' **in node_list:**
Check if S dominates S' by using propositions from 1-6
If partial schedule S is dominant to S'
1. Add partial schedule S to node-list set.
2. Remove partial schedules S’ from node_list set
Elif partial schedule S' is dominant:
1. Do not add partial schedule S to node-list set.
Else:
1. Add partial schedule S to node-list set.
Else:
1. Add partial schedule S to node-list set.

End
Output node-list

7. Computational results

Experiments were conducted using various parameter combinations and different sample sizes to evaluate the performance of the developed algorithms.

Experimental Environment:
The algorithms were implemented in JetBrains PyCharm Professional 2021.3.2, and experiments were conducted on a personal computer equipped with a 3.2 GHz 6-Core Intel Core i7-8700 processor and 16 GB of memory. The operating system used for these experiments was Windows 10 Pro 21H1.

Data Generation:
Job processing times and due dates were generated randomly, with processing times drawn from a uniform distribution over the integers (0,100) and due dates from the interval [1, 5].

Experimental Setup:
The experiments encompassed various parameter combinations and sample sizes. The total number of jobs assigned to the agents was set at 5, 10, and 15. Two deterioration coefficients (β) (0.1 and 0.2) and three learning coefficients (α)(70%, 80%, and 90%) were considered. This resulted in a total of 30 distinct parameter combinations. Each combination was tested five times, yielding a total of 90 experiments.

The experiments provided three primary metrics for evaluation: nodes-ratio, the error-percentage, and the CPU-time.

The nodes-ratio metric indicates the extent to which the developed algorithm reduces the number of generated nodes compared to the classical branching method. It is calculated using Equation 10 where TN and TN* are denoted as the nodes generated by the classical branch-and-bound algorithm and the total nodes generated by the proposed algorithm, respectively:

$$Nodes - ratio = TN^{*}/TN$$

The error percentage quantifies the percentage difference between the total weighted completion time of the initial solution and the solution obtained by the branch-and-bound algorithm. Calculation involves Equation 11 where OFV and OFV^{*} are denoted as the total weighted completion time of the initial solution and the branch-and-bound algorithm solution, respectively:

$$Error - percentage = (OFV - OFV^{*})/OFV$$

CPU time represents the duration in seconds required for processing the developed algorithm. Note that CPU times are influenced not only by the algorithm's efficiency but also by the coding implementation.

The efficiency of the proposed methodology hinges on several critical factors, including the total number of jobs, the learning coefficient, and the deterioration coefficient. These factors collectively shape the performance of the algorithm and its ability to provide optimal solutions. The assessment of the methodology's effectiveness for different parameter values is summarized in Table 1.

A comprehensive analysis of the relationships between these parameters and key results, such as processing time and the number of nodes, reveals notable trends. Specifically, as the total number of jobs increases, both processing time and the error percentage exhibit a corresponding increase. These relationships are visually represented in Figure 5and Figure 6. On the other hand, although the number of nodes increases as the number of jobs increases, the value of the nodes-ratio decreases as shown in Figure 4.

Table 1. Performance Indicators for Proposed Algorithms for the Second Problem

	α	β	Total number of jobs (n)		
			5	10	15
Node – ratio (%)	70%	0,2	2,974359	0,0113198	2,34E-06
	70%	0,1	3,025641	0,0100709	6,88E-06
	80%	0,2	3,025641	0,0076175	2,96E-07
	80%	0,1	3,2820513	0,0078	2,75E-06
	90%	0,2	2,5128205	0,0045519	7,47E-08
	90%	0,1	2,5128205	0,0069951	4,09E-07
	MEAN		2,8888889	0,0080592	2,12E-06

Error Percentage	70%	0,2	0,0252641	0,0559249	0,0623773
	70%	0,1	0,0321291	0,0428	0,0428416
	80%	0,2	0,0121311	0,0420808	0,0584301
	80%	0,1	0,0139486	0,0251013	0,0244233
	90%	0,2	0,0036245	0,0560238	0,0497368
	90%	0,1	0,0054943	0,0145459	0,028879
	MEAN		0,015432	0,0394128	0,047204
CPU time	70%	0,2	0,03019	9,5659109	2300,0308
	70%	0,1	0,034335	7,757144	5326,2117
	80%	0,2	0,027825	4,272038	346,22927
	80%	0,1	0,0476699	4,802578	2728,5332
	90%	0,2	0,0176551	2,3049759	79,404798
	90%	0,1	0,0305834	5,205898	411,402
	MEAN		0,0313764	5,6514242	1863,4005

When considering higher values of the learning effect coupled with lower values of the deterioration effect, an interesting set of consequences emerges. Specifically, as the learning effect becomes more pronounced and the deterioration effect diminishes, the processing times of jobs are expected to shorten. Consequently, this results in an expansion of slack time within the scheduling framework. Additionally, the number of jobs that can be accommodated and processed during these slack-time intervals is poised to increase, CPU time tends to increase, reflecting the algorithm's increased computational demands, while the node ratio also increases, indicating a less efficient use of computational resources.

Conversely, when dealing with lower values of the learning effect combined with higher values of the deterioration effect, a different set of outcomes unfolds. In this scenario, as the learning effect diminishes and the deterioration effect intensifies, job processing times tend to lengthen. Consequently, the available slack time is reduced, limiting the number of jobs that can be accommodated and processed during these constrained intervals. As a result, CPU time and node ratio tend to decrease.

These observations underscore the intricate relationship between the learning and deterioration effects on job processing times and their significant impact on various performance metrics, offering valuable insights for optimizing the scheduling algorithm under different conditions.

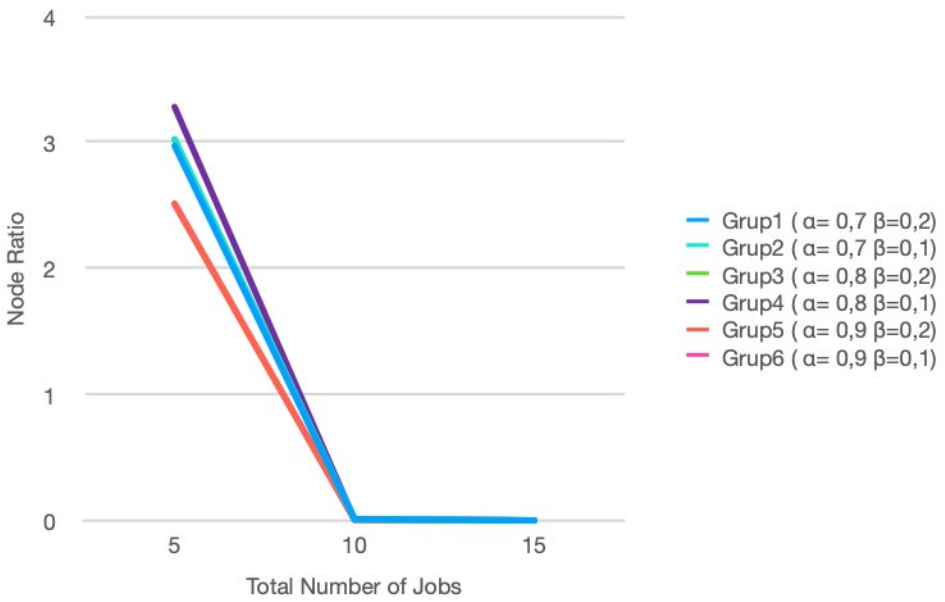


Figure 4. Relation Between Total Number of Jobs and Nodes Ratio

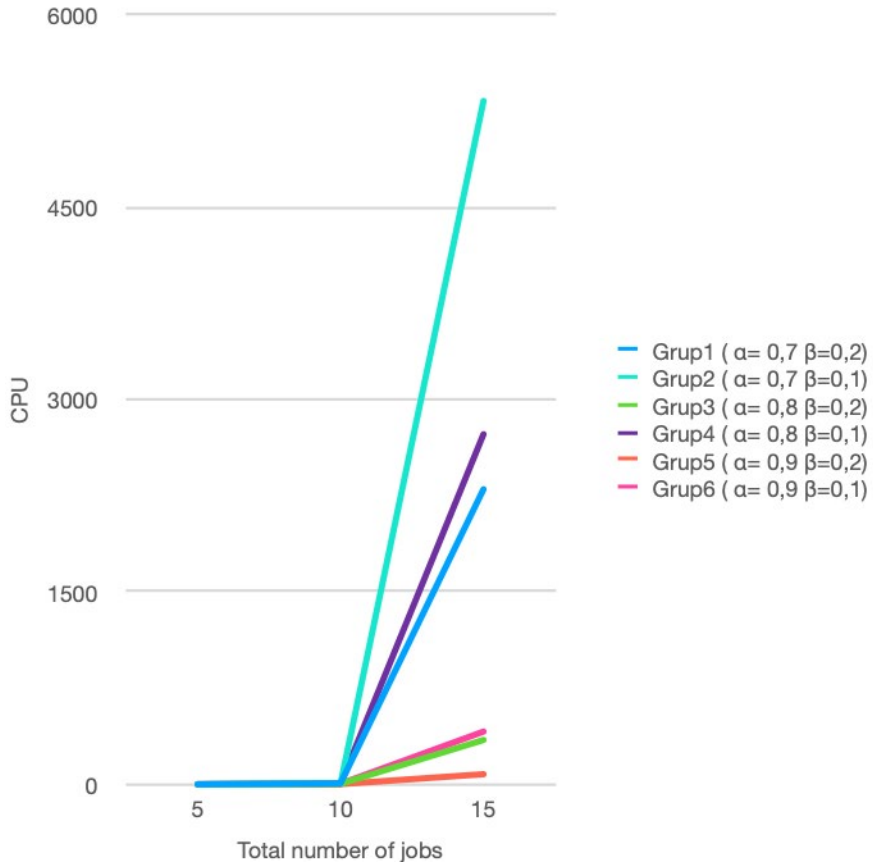


Figure 5. Relation Between Total Number of Jobs and CPU time

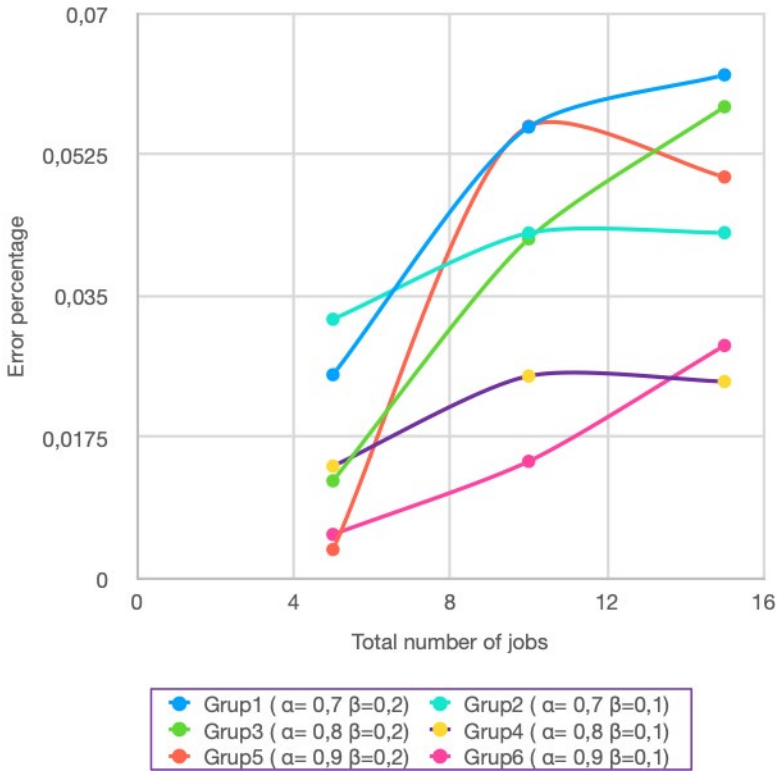


Figure 6. Relation Between Total Number of Jobs and Error Percentage

8. Conclusion

This paper presented a methodology designed to address the challenging multi-agent scheduling problem involving jobs subject to the simultaneous influence of learning and deterioration effects. The primary objective of this problem was to minimize the total weighted completion time for all agents while enforcing the constraint that no tardy jobs were permissible for the second agent.

The proposed methodology was structured into two key phases: an initial heuristic for generating an initial solution and a subsequent branch-and-bound algorithm for refining and optimizing the solution further. To enhance the efficiency of the branch-and-bound algorithm and reduce its computational complexity, a set of dominance rules and pruning rules were introduced.

The computational results showcased the effectiveness of the branch-and-bound algorithm, particularly when coupled with the initial heuristic solution. Notably, this approach demonstrated commendable performance in terms of the number of nodes generated and execution time across a range of experiments. The methodology was rigorously tested with up to 15 jobs, revealing that its success was notably influenced by factors such as a smaller learning coefficient, a larger deterioration coefficient, and reduced slack time intervals for jobs belonging to agent B.

In summary, this study offers a promising methodology for tackling complex multi-agent scheduling problems characterized by the simultaneous influence of learning and deterioration effects. The proposed approach, combining heuristic and branch-and-bound techniques, provides a robust foundation for efficiently solving such problems, with its performance particularly favorable under certain parameter conditions.

References

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., and Pacifici, A., Nondominated schedules for a job-shop with two competing users. Comput. Math. Organ. Theory, 191-217, 2000.

Agnetis, A., Mirchandani, P. B., Pacciarelli, D., and Pacifici, A., Scheduling problems with two competing users. UniversitaÁ di Roma "Tor Vergata," Centro Vito Volterra, Working Paper No. 452, 2001.

Baker, K. R. and Smith, J. C., A multiple-criterion model for machine scheduling. Journal of Scheduling, 6(1): 7-16, 2003.

Cheng, T. C. E., Chung, Y. H., Liao, S. C., and Lee, W. C., Two-agent single-machine scheduling with release times to minimize the total weighted completion time. Computers and Operations Research, 40: 353-361, 2013.

Cheng, T. C. E., Wu, W. H., Cheng, S. R., & Wu, C. C., Two-agent scheduling with position-based deteriorating jobs and learning effects. Applied Mathematics and Computation, 217(21), 8804–8824, 2011.

Choi, B. C. and Chung, J., Two-agent single-machine scheduling problem with just-in-time jobs. Theoretical Computer Science, 543: 37–45, 2014.

Danacı, T., Toksari, M.D., "A branch-and-bound algorithm for two-competing-agent single-machine scheduling problem with jobs under simultaneous effects of learning and deterioration to minimize total weighted completion time with no-tardy jobs", International Journal of Industrial Engineering, Vol. 28 No. 6, 2021.

Lee, K., Choi, B. C., Leung, J. Y. T., and Pinedo, M. L., Approximation algorithms for multi-agent scheduling to minimize total weighted completion time. Information Processing Letters, 109(16): 913–917, 2009.

Lee, W. C. and Wang, J. Y., A scheduling problem with three competing agents. Computers and Operations Research, 51: 208–217, 2014.

Lee, W.C., Wang, W.J., Shiau, Y.R., & Wu, C.C., A single-machine scheduling problem with two-agent and deteriorating jobs. Applied Mathematical Modelling, 34, 3098–3107, 2010.

Nong, Q. Q., Cheng, T. C. E., & Ng, C. T., Two-agent scheduling to minimize the total cost. European Journal of Operational Research, 215(1), 39–44, 2011.

Soltani, R., Jolai, F., and Zandieh, M., Two robust meta-heuristics for scheduling multiple job classes on a single machine with multiple criteria. Expert Systems with Applications, 37(8): 5951–5959, 2010.

Wu, C.C., Lee, W.C., & Liou, M.J., Single-machine scheduling with two competing agents and learning consideration. Information Sciences, 251, 136–149, 2013.

Wu, W. H., Cheng, S. R., Wu, C. C., & Yin, Y., Ant colony algorithms for a two- agent scheduling with sum-of processing times-based learning and deteriorating considerations. Journal of Intelligent Manufacturing, 23(5), 1985–1993, 2012.

Wu, W.H., Solving a two-agent single-machine earning scheduling problem. International Journal of Computer Integrated Manufacturing, 27, 20-35, 2014.

Yin, Y., Wu, W. H., Cheng, T. C. E., Wu, C. C., and Wu, W. H., A honey-bees optimization algorithm for a two-agent single-machine scheduling problem with ready times. Applied Mathematical Modelling, 39: 2587–2601, 2015.

Zhang, Y., Yuan, J., Ng, C., and Cheng, T. C., Pareto-optimization of three-agent scheduling to minimize the total weighted completion time, weighted number of tardy jobs, and total weighted late work. Naval Research Logistics, 68: 378-393, 2020.

Biography

Tugba Danaci is an accomplished industrial engineer and production management consultant with a diverse educational background. She pursued a double major in Management Engineering and Mechanical Engineering, specializing in System Dynamics and Control during her undergraduate studies at Istanbul Technical University. Her academic journey continued with a Master of Science in Strategic Management in Defense Technologies and culminated in a Ph.D. in Industrial Engineering from Cleveland State University and Erciyes University. Alongside her academic studies, she has held various consulting and managerial roles within companies operating in the defense industry. Her focus includes the implementation of AS/EN 9100 Rev D quality management systems, digitalization and optimization of production processes, their migration to cloud platforms, bespoke software development tailored to specific company needs, and spearheading projects ensuring compliance with NATO and National standards for facility security.