ACCSCIENCE PUBLISHING

An International Journal of Optimization and Control: Theories & Applications
ISSN: 2146-0957 eISSN: 2146-5703
Vol.16, No.4, pp.1586-1604 (2026)
https://doi.org/10.36922/IJOCTA026140054

IJOCTA

RESEARCH ARTICLE

Minimizing total actual flow time for multi-job batch scheduling on identical machines with a common due date

Rinto Yusriski[1]*, Andri Rachmat Kumalasian Nasution[1], and Mohd Azlan Suhaimi[2]

[1]Department of Industrial Engineering, Faculty of Manufacturing Technology, Universitas Jenderal Achmad Yani, Bandung, Indonesia

[2]Advanced Manufacturing Research Group, Faculty of Mechanical Engineering, Universiti Teknologi Malaysia, Skudai, Johor, Malaysia

rinto.yusriski@lecture.unjani.ac.id, andri.rachmatk@lecture.unjani.ac.id, azlansuhaimi@utm.my

ARTICLE INFO

Article history:
Received: April 5, 2026
Revised: June 2, 2026
Accepted: June 9, 2026
Published Online: July 8, 2026

Keywords:
Batch scheduling
Parallel machines
Branch-and-Bound
Common due date
Total actual flow time

AMS Classification 2010:
90B35, 90C57, 90C10

ABSTRACT

Batch-processing on identical parallel machines is widely used in discrete manufacturing to reduce setup frequency and maintain coordinated material flow. This paper studies a discrete multi-job batch scheduling problem with job-dependent setup times and the objective of minimizing total actual flow time. Several jobs with integer demands must be assigned to parallel machines and completed by a single common due date. The scheduling decisions include assigning jobs to machines, determining the number and batch sizes, and sequencing the resulting batches under a backward just-in-time scheduling approach. An exact Branch-and-Bound (B&B) algorithm is developed for this problem. Feasible solutions are evaluated using a backwards-packed scheduling procedure, while horizon-based feasibility screening and an allocation-level lower bound are used to prune the search tree. A numerical example and computational experiments show that the algorithm can find globally optimal solutions for small- to medium-sized cases. This makes the proposed B&B a useful exact benchmark for future heuristic development.

1. Introduction

Batch-oriented production is widely used in discrete manufacturing systems such as heat treatment, coating, textile dyeing and finishing, and molding operations, where several units of the same job are processed together to reduce setup frequency and stabilize material flow.[1] In parallel machine environments, production planners must jointly determine job-to-machine allocation, integer batch formation, and batch sequencing because these decisions affect workload balance, setup frequency, and batch start times.[2–4] Recent studies on smart and sustainable production also emphasize the importance of work-in-process (WIP) control, resource utilization, and coordinated production timing.[5–7] In make-to-order and shipment-driven systems, all jobs are often required to meet a single common due date. Under this condition, backward or just-in-time (JIT) scheduling is relevant because production is scheduled as late as possible while still satisfying the due date requirement. Total actual flow time is therefore an appropriate performance measure because it evaluates the cumulative time that all units remain on the shop floor until the due date,[8] reflecting WIP reduction and due date-oriented production control.

Despite extensive research on batch scheduling and parallel machine scheduling, existing studies have primarily addressed batch processors, flow-shop batching, integrated production–delivery batching, additive manufacturing, incompatible job families, release dates, maintenance, energy

*Corresponding Author

consumption, and resource constraints.[8–35] However, the simultaneous integration of job-to-machine allocation, integer batch formation, inter-job sequencing, job-dependent setup times, backward scheduling, a single common due date, and total actual flow time minimization in an identical parallel machine environment remains underexplored.

To address this gap, this study develops a discrete multi-job batch scheduling model on identical parallel machines under a single common due date. The objective is to minimize total actual flow time within a backward JIT scheduling framework. An exact Branch-and-Bound (B & B) algorithm is proposed to systematically enumerate allocation, backward sequencing, and integer batch-size decisions, while incorporating horizon-based feasibility screening and an allocation-level lower bound (LB) to reduce the search space and certify global optimality for small- to medium-sized benchmark instances.

The main contributions of this study are threefold. First, it formulates an integrated discrete-batch scheduling model that jointly considers allocation, integer batching, sequencing, job-dependent setup times, and a single common due date on identical parallel machines. Second, it adapts the total actual flow time criterion to a backward JIT parallel machine batching environment. Third, it develops an exact B & B benchmark procedure and provides numerical analyses that clarify the interaction among workload allocation, setup consumption, batch splitting, and backward schedule positioning.

Figure 1 summarizes the study's conceptual framework by linking the production context, integrated scheduling decisions, backward JIT logic, tailored B & B procedure, schedule evaluation, and optimal benchmark solution.

As shown in **Figure 1**, the proposed framework links the batch-oriented production system, input parameters, integrated scheduling decisions, and exact B & B solution procedure. Job demand, processing time, setup time, number of machines, and due date define the allocation, batching, and sequencing decisions. These decisions are evaluated using backward JIT scheduling, in which batches are scheduled as late as possible before the common due date. The B & B procedure explores the feasible decision space through allocation, sequencing, and batch-size branching, supported by horizon-feasibility screening and an allocation-level LB. Each feasible leaf is evaluated using a backwards-packed schedule, and the solution with the minimum total actual flow time is selected as the optimal benchmark solution.

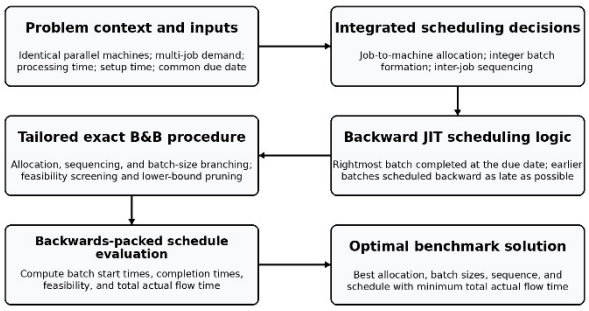

Figure 1. Conceptual framework of the proposed multi-job batch scheduling model on identical parallel machines under a single common due date
Abbreviations: B & B: Branch-and-Bound; JIT: Just-in-time.

2. Literature review

Batch scheduling has received considerable attention because batching decisions directly affect setup frequency, machine utilization, WIP inventory, and due date performance. In batch-discrete manufacturing systems, production orders are grouped into batches to improve operational efficiency and stabilize production flow.[1] Recent Industry 4.0 production environments further require scheduling models that can coordinate machine assignment, task allocation, and shop-floor control decisions.[2] In make-to-order systems with parallel resources, lot-sizing and scheduling decisions are often complicated by machine eligibility, capacity limitations, extra shifts, and backorders.[3] In addition, production uncertainty has motivated extensions of dispatching and scheduling rules to improve robustness under variable processing conditions.[4] These developments indicate that modern batch scheduling is not only concerned with machine utilization, but also with controlling WIP accumulation, coordinating production timing, and maintaining system responsiveness.

This broader concern has also been emphasized in recent studies on smart and sustainable production, where WIP control, resource utilization, and production timing are treated as important performance dimensions. Dey *et al.*[5] developed an autonomation policy to control WIP inventory in a smart production system, demonstrating that production–control policies should explicitly account for inventory accumulation and system responsiveness. Sarkar

et al.[6] examined sustainable smart multi-type biofuel manufacturing with optimal energy utilization under flexible production, indicating that production flexibility and energy-aware decision-making are increasingly important in modern manufacturing. Sarkar *et al.*[7] further studied the sustainable smart production of electric vehicles, considering the WIP inventory of outsourced spare parts, and highlighted the interactions among production planning, outsourced components, WIP inventory, and sustainability-oriented performance. These studies provide a broader perspective on smart and sustainable production that complements the scheduling focus of the present research. In particular, they reinforce the importance of controlling WIP, coordinating production timing, and improving resource utilization, which are also central concerns in total actual flow time minimization under a JIT scheduling environment.

In this context, total actual flow time becomes a particularly relevant performance measure because it directly evaluates how long parts remain in the production system before their due dates. Halim *et al.*[8] introduced total actual flow time as a JIT performance measure that evaluates the cumulative residence time of parts in the shop. This criterion differs from classical completion-time or tardiness-based measures because it focuses on the actual time items spend in the production system rather than solely on their completion times or lateness. Subsequent studies extended this concept to batch-processing environments, incorporating additional setup requirements, inspection policies, learning and forgetting effects, deterioration, and multi-job structures.[9–13] These studies show that total actual flow time is suitable for evaluating JIT production systems because it penalizes early release and excessive WIP accumulation.

The setup–splitting trade-off is also central in batch scheduling. Splitting a job into more batches may provide scheduling flexibility, but it also increases the number of setups and may force some batches to start earlier. Integrated production–delivery batching studies indicate that batching decisions influence both production timing and downstream logistics.[14] Similar complexity appears in additive manufacturing, where order acceptance and batching decisions must be coordinated to manage production capacity efficiently.[15] Overlapping and flow-based production strategies can improve responsiveness, but they still require careful coordination among batching, sequencing, and due date constraints.[16,17] Therefore, batch-number, batch-sizing, and batch scheduling decisions should be treated jointly rather than independently.

Several recent studies have expanded batch scheduling models into flow-shop and multi-stage production environments. Kurniawan *et al.*[18] considered flow-shop batch scheduling with pre-processing and time-changing effects to minimize total actual flow time. Sibarani *et al.* [19,20] studied multi-item, multi-stage flow-shop batch scheduling models with dedicated machines and batch-job-batch processors. Other related extensions have examined hybrid flow-shop structures with unique and common components, heterogeneous job and batch processors, and two-machine flow-shop batch scheduling under total actual flow time objectives.[21–23] These studies demonstrate the continued relevance of total actual flow time in multi-stage production systems. However, their focus is primarily on flow-shop configurations rather than on identical parallel machine environments with simultaneous allocation, batching, and sequencing decisions.

Parallel machine batch scheduling has also been widely studied. Recent models have considered incompatible job families, release dates, tardiness penalties, machine availability, additive resource assignment, maintenance policies, energy efficiency, setup effects, and integrated order allocation.[24–35] For example, population-based local search has been used for parallel machine batch scheduling with incompatible job families and release dates,[24] while other studies have addressed fast parallel scheduling in steel cold rolling,[25] additive resource assignment and machine availability,[26] integrated batch production and supplier-related decisions,[27] and production–maintenance integration.[28] Energy-efficient scheduling and flexible preventive maintenance have also been incorporated into unrelated parallel batch-processing machines.[29] In addition, mixed-model parallel assembly line problems have been addressed through integrated order allocation, lot-sizing, and sequencing decisions.[30] More recent studies further confirm the continuing relevance of parallel machine and parallel-batch scheduling. ElWakil *et al.*[31] addressed batch scheduling on non-identical parallel machines using LBs, mixed-integer programming, branch-and-price, and heuristic procedures. Santos *et al.*[32] developed an adaptive local search for large-scale parallel machine scheduling in textile production that accounts for release dates and sequence-dependent setup

times. Ou and Li[33] studied mixed-batch machines with inclusive processing–set restrictions and non-identical capacities. Sun *et al.*[34] extended generalized parallel batch scheduling to consider batch-processing times that are nondecreasing in job width and thickness. Fan *et al.*[35] examined a scheduling game on parallel batch machines with setup cost. These contributions show that allocation, batching, sequencing, setup, and capacity decisions remain highly interdependent in parallel machine systems.

Just-in-time and common due date scheduling are closely related to the present study. JIT material-handling scheduling has been examined using hybrid branch-and-price methods in mixed-model assembly lines.[36] Common due date and due date assignment problems have also been studied in identical or unrelated parallel machine environments.[37–42] Related scheduling studies have further considered distributed additive manufacturing with carbon-emission concerns, indicating that scheduling models are increasingly extended to broader sustainability-oriented manufacturing settings.[43] These studies confirm that due date coordination, production timing, and sustainability-related scheduling considerations are critical issues. However, most of them emphasize completion time, earliness–tardiness, job rejection, due date deviation, or carbon-related objectives rather than total actual flow time under a backward scheduling logic.

Exact optimization and advanced heuristic methods have been developed for several variants of parallel machine and batch scheduling. Constraint programming has been used for parallel batch scheduling with incompatible job families,[44] while reinforcement learning combined with simulated annealing has been proposed for setup-driven parallel batch scheduling.[45] Structural properties of batch scheduling on parallel machines have also been investigated.[46] Model-based branch-and-price algorithms and decomposition-based formulations have proven effective for related parallel machine batching and scheduling problems.[47–49] These studies indicate that exact and hybrid methods are useful for producing benchmark solutions, although their computational burden generally increases rapidly with problem size.

A direct basis for this study is the unrelated parallel machine batch scheduling model with resource constraints and sequence-dependent setup times under a total actual flow time objective.[50] However, that model considers multiple due dates and unrelated machines. The present study differs in that it focuses on identical parallel machines under a single common due date. This common due date setting intensifies capacity competition within a shared time horizon and strengthens the interaction among allocation, batching, and sequencing decisions. Therefore, a dedicated model and exact benchmark algorithm are needed.

Based on the reviewed literature, the research gap can be more precisely stated. Previous studies have addressed batch scheduling, parallel machine scheduling, smart production, WIP control, energy-aware scheduling, total actual flow time, due date coordination, sustainability-related scheduling, and exact or hybrid optimization methods from different perspectives. However, the integrated problem of multi-job integer batch scheduling on identical parallel machines with job-dependent setup times, job-to-machine allocation, inter-job sequencing, backward JIT scheduling, and total actual flow time minimization under a single common due date remains insufficiently explored. This study addresses this gap by formulating the problem as an integrated discrete scheduling model and solving it using a tailored exact B & B procedure with horizon-feasibility screening, allocation-level LB, and backwards-packed schedule evaluation.

To clarify the novelty of the proposed model, **Table 1** maps the main research streams against the key decision elements considered in this study. The table highlights which aspects have been addressed in previous studies and, more importantly, identifies the unaddressed combination of allocation, integer batching, sequencing, setup times, backward JIT scheduling, a single common due date, and total actual flow time minimization that is addressed in the present work.

As summarized in **Table 1**, previous studies have addressed important aspects of batch scheduling, including total actual flow time, JIT scheduling, flow-shop and multi-stage batching, parallel machine scheduling, common due date coordination, smart and sustainable production, and exact or hybrid optimization methods. However, these elements have generally been studied separately or only partially integrated. The present study addresses this gap by jointly considering job-to-machine allocation, integer batch formation, inter-job sequencing, job-dependent setup times, backward JIT scheduling, a single common due date, and total actual flow time minimization in an identical parallel machine environment. Accordingly, the proposed

Table 1. Gap-based novelty positioning of the present study

Research stream	**Main coverage in previous studies**	**The remaining gap addressed by this study**
Batch-oriented production, Industry 4.0 scheduling, and production uncertainty[1–4]	Previous studies emphasized batch-discrete manufacturing, machine assignment, lot-sizing, production scheduling, and robustness under uncertain processing conditions	These studies provide the production context, but do not jointly address integer batching, inter-job sequencing, backward JIT scheduling, and total actual flow time minimization
Smart and sustainable production[5–7]	Prior works highlighted WIP control, resource utilization, energy-aware production, smart manufacturing, and sustainability-oriented production planning	These studies support the WIP-oriented motivation, but do not formulate a detailed parallel machine batch scheduling model using total actual flow time as the objective
Total actual flow time and JIT batch scheduling[8–13]	This stream established total actual flow time as a JIT-oriented measure and extended it to single-machine, batch-processor, inspection, learning, forgetting, deterioration, and multi-job settings	Existing models mainly focus on non-parallel machine settings and do not integrate allocation, batching, and sequencing on identical parallel machines under a single common due date
Production–delivery, additive manufacturing, overlapping, and flow-based batching[14–17]	These studies showed that batching decisions influence production timing, delivery coordination, order acceptance, overlapping operations, and production responsiveness	They do not solve the integrated identical parallel machine batch scheduling problem under backward JIT logic and total actual flow time minimization
Flow-shop and multi-stage batch scheduling[18–23]	Previous works developed total actual flow time models for flow shop, hybrid flow shop, and multi-stage batch-processing systems	These studies focus on sequential multi-stage systems, whereas the present study addresses the parallel machine counterpart involving allocation, batching, and sequencing within a shared due date horizon
Parallel machine batch scheduling[24–35]	Prior research addressed parallel batching, setup effects, release dates, resource constraints, machine availability, maintenance, energy efficiency, and heuristic or exact solution methods	Most studies do not combine total actual flow time, backward JIT logic, job-to-machine allocation, integer batch formation, inter-job sequencing, and a single common due date in one model
JIT, common due date, and sustainability-oriented scheduling[36–43]	This stream covered JIT material handling, common due date coordination, due date assignment, earliness–tardiness, online allocation, and carbon-emission considerations	These studies generally focus on completion time, due date deviation, earliness–tardiness, job rejection, online allocation, or carbon-related objectives rather than total actual flow time
Exact and hybrid optimization methods[44–49]	Previous studies developed constraint programming, reinforcement-learning-based hybrid search, structural analysis, branch-and-price, and decomposition-based formulations	They demonstrate the relevance of exact and hybrid methods, but do not provide an exact B&B benchmark for the integrated problem addressed in this study
Closest prior parallel machine model using total actual flow time[50]	The closest related model considered unrelated parallel machines, resource constraints, sequence-dependent setups, multiple due dates, and total actual flow time minimization	The present study differs in that it focuses on identical parallel machines, a single common due date, backward JIT scheduling, and exact benchmarking of allocation, sequencing, and integer batching decisions

(*Cont'd...*)

Table 1. (*Continued*)

Research stream	Main coverage in previous studies	The remaining gap addressed by this study
Present study	This study integrates job-to-machine allocation, integer batching, inter-job sequencing, job-dependent setup times, backward JIT scheduling, horizon-feasibility screening, allocation-level LB, and exact B&B evaluation	It provides an exact benchmark for discrete multi-job batch scheduling to minimize total actual flow time on identical parallel machines under a single common due date

Abbreviations: B&B: Branch-and-Bound; JIT: Just-in-time; LB: Lower bound; WIP: Work-in-process.

model provides a unified discrete scheduling framework and an exact B & B benchmark for future heuristic and hybrid methods.

3. Methods

This section presents the methodology for solving the discrete multi-job batch scheduling problem on identical parallel machines with a single common due date. The methodology consists of two parts. First, the total actual flow time objective and the mathematical model are formulated under backward JIT scheduling. Second, an exact B & B procedure is developed to optimally solve the discrete model. The B & B enumerates allocation, backward sequencing, and integer batch-size decisions, and evaluates each complete solution using a backward-packed scheduling rule consistent with the total actual flow time definition.[8]

We consider a manufacturing system with J jobs processed on M identical parallel machines under a single common due date d. Each job j has an integer demand n_j, a unit processing time t_j, and a job-dependent setup time s_j. A setup of duration s_j must be performed immediately before processing each batch of job j. Machines are single-capacity, batches are non-preemptive, and batches from different jobs may be interleaved on the same machine. Scheduling follows a backward JIT logic: On each machine, the job block processed closest to the due date is placed last, and earlier work is scheduled backward as late as feasibility allows.

3.1. Notation (indices, parameters, and variables)

Indices : Jobs $j = 1, 2, ..., J$; machines $m = 1, 2, ..., M$; batch indices $i = 1, 2, ..., N_{j,m}$ where $i = 1$ denotes the batch closest to the due date within each (j, m) block and indices increase leftward.

Parameters : n_j (integer demand of job j.); t_j (unit processing time of job j); s_j (setup time required immediately before any batch of job j); and d (single common due date).

Decision variables : $A_{j,m}$ (integer allocation of job j to machine m); $Q_{j,m,i}$ (integer size of batch i of job j on machine m); $B_{j,m,i}$ (start of processing after setup); $C_{j,m,i}$ (completion time); and π_m (backward job order on machine m).

Auxiliary quantity : $N_{j,m}$ is the number of realized (nonzero) batches for job j on machine m, i.e., $N_{j,m} = |\{i : Q_{j,m,i} > 0\}|$.

3.2. Performance measure and objective function

Following previous work[8], the performance measure is the total actual flow time, which evaluates the cumulative time that all units remain on the shop floor until the common due date. For a batch of size $Q_{j,m,i}$ that starts processing at the time $B_{j,m,i}$, the batch contribution is:

$$F^a_{j,m,i} = \left(d - B_{j,m,i}\right)Q_{j,m,i} \tag{1}$$

The total actual flow time is

$$F^a = \sum_{j=1}^{J}\sum_{m=1}^{M}\sum_{i=1}^{N_{j,m}}\left(d - B_{j,m,i}\right)Q_{j,m,i} \tag{2}$$

Since d and $\sum_{j=1}^{J} n_j$ are fixed, **Equation 2** can be rewritten as

$$F^a = d\sum_{j=1}^{J} n_j - \sum_{j=1}^{J}\sum_{m=1}^{M}\sum_{i=1}^{N_{j,m}} B_{j,m,i}\ Q_{j,m,i} \tag{3}$$

Thus, minimizing total actual flow time is equivalent to maximizing the weighted sum of batch start times. Under backward scheduling, the start time of a batch is determined by its completion time and processing requirement:

$$B_{j,m,i} = C_{j,m,i} - t_j Q_{j,m,i} \tag{4}$$

3.3. Mathematical model

The mathematical model is formulated as

$$\min F^a \tag{5}$$

subject to

$$\sum_{m=1}^{M} A_{j,m} = n_j, \quad \forall j; \tag{6}$$

$$\sum_{i=1}^{N_{j,m}} Q_{j,m,i} = A_{j,m}, \quad \forall j,m; \tag{7}$$

$$C_{j,m,i} = B_{j,m,i} + t_j Q_{j,m,i}, \qquad \forall j,m,i; \tag{8}$$

$$B_{g,m,N_{g,m}} = C_{h,m,1} + s_g, \quad \forall m, \forall (g,h) \text{ adjacent on } \pi_m \tag{9}$$

$$C_{\pi_m(1),m,1} = d, \quad \forall m; \tag{10}$$

$$\sum_{j=1}^{J}\left(t_j A_{j,m} + s_j N_{j,m}\right) \le d, \qquad \forall m; \tag{11}$$

$$A_{j,m} \in \mathbb{Z}_{\ge 0},\ Q_{j,m,i} \in \mathbb{Z}_{>0},\ B_{j,m,i} > 0,\ C_{j,m,i} > 0; \tag{12}$$

Equation 5 defines the objective: to minimize the total actual flow time. Constraints in **Equations 6–7** ensure that job demands are fully allocated across machines and that each allocation is exactly partitioned into integer batches. Constraint in **Equation 8** defines the completion time of each batch. Constraint in **Equation 9** indicates that if the job block g is scheduled later than h, or closer to the common due date on machine m under backward scheduling, then the start time of the first batch of the job g is determined from the finish time of the last batch of the job h plus the setup time of the job g. Constraint in **Equation 10** ensures that the rightmost batch of the rightmost job block on each machine completes at the common due date. Constraint in **Equation 11** imposes horizon-feasibility over $[0,d]$, where $N_{j,m}$ is the number of realized batches. Constraint in **Equation 12** enforces integrality.

3.4. Exact Branch-and-Bound procedure (step-by-step)

Because allocation, backward sequencing, and integer batch-size decisions are strongly interdependent, the model is combinatorial. Therefore, an exact B & B procedure is developed as a benchmark solution method. The B & B explores the feasible decision space using the stepwise branching scheme described below.

Step 0. Initialization: Set the incumbent objective value upper boundary (UB) ← $+\infty$.

Initialize the best solution record as empty.

Step 1. Allocation branching and early screening: Enumerate all feasible integer allocations $A_{j,m}$ satisfying constraint in **Equation 6**. For each allocation, perform early feasibility screening using the minimum-batch requirement: Any positive allocation must be processed in at least one batch. Define $N_{j,m}^{min}=1$ if $A_{j,m}>0$, and $N_{j,m}^{min}=0$ otherwise, and compute the minimum workload on each machine $W_m^{min}=\sum_{j=1}^{J}\left(t_j A_{j,m}+s_j N_{j,m}^{min}\right)$. If $W_m^{min}>d$ for any machine m, the allocation is infeasible and discarded. Next, compute the optimistic allocation-based lower bound (LB[A]) used for pruning:

$$LB(A)=\sum_{j=1}^{J}\sum_{m=1}^{M}t_j A_{j,m}^2 \tag{13}$$

If $LB(A)\geq UB$, discard the allocation-based estimate because it cannot produce an improved feasible solution.

Step 2. Backward sequencing branching: For each retained allocation, determine the set of jobs assigned to each machine. Enumerate all backward job orders π_m on each machine (only machines with more than one assigned job require sequencing).

Step 3. Integer batch-size branching: For each combination of allocation A and backward job orders π, enumerate integer batch-size compositions $Q_{j,m,i}$ for each positive allocation $A_{j,m}$ such that the constraint in **Equation 7** holds. To reduce symmetric solutions, batch sizes within each (j,m) blocks are generated in non-increasing order.

Step 4. Leaf evaluation by backwards-packed scheduling: Once $\left(A_{j,m},\pi_m,Q_{j,m,i}\right)$ are fixed, a complete schedule is constructed by backward packing. For each machine, the rightmost job block is anchored at the common due date. Within each job block, batches are placed backward without inserted idle time, and each batch is immediately preceded by its setup. After a job block is placed, the next (left) job block is positioned so that the inter-job relation in **Equation 9** is satisfied tightly (i.e., no idle time is inserted between adjacent job blocks). If any computed processing start time is negative, the solution violates the horizon and is discarded; otherwise, it is feasible.

Step 5. Objective evaluation and incumbent update: For any feasible complete solution, compute F^a using **Equation 2**. If $F^a<\mathrm{UB}$, update $UB\leftarrow F^a$ and store the corresponding $\left(A_{j,m},\pi_m,Q_{j,m,i}\right)$ as the current best solution.

Step 6. Termination and optimality: **Steps 1–5** are repeated until all combinations generated by the stepwise branching scheme have been either evaluated or discarded by feasibility screening or LB dominance. When the procedure terminates, the incumbent solution is a global optimum of the discrete model.

3.5. Data source and computational instance design

The proposed model and algorithm were evaluated using constructed benchmark instances, as the study aims to develop an exact solution benchmark rather than estimate parameters from a specific industrial dataset. The illustrative example provides a traceable case for explaining the B & B procedure, while the scalability experiments use controlled instances to examine allocation, batching, sequencing, setup effects, feasibility screening, and LB pruning under reproducible conditions. To support transparency, the generated test instances and computational results are available on Zenodo at https://doi.org/10.5281/zenodo.20503546 (record number 20503546).

4. Results

This section presents two sets of numerical results. First, an illustrative example demonstrates the step-by-step implementation of the proposed exact B & B algorithm, including allocation screening, backward sequencing, integer batching, and backwards-packed schedule evaluation. Second, computational experiments are conducted to evaluate the scalability, feasibility screening, LB pruning, and runtime performance of the proposed algorithm. The numerical example and computational experiments are based on the constructed benchmark instances described in Section 3.5. Throughout this section, the total actual flow time is denoted by F^a.

4.1. Illustrative example

Consider $J=2$ jobs processed on $M=2$ identical parallel machines under a single common due date $d=14$. Job data are:

- Job 1: Demand $n_1=3$ units, processing time $t_1=2$, setup $s_1=3$.
- Job 2: Demand $n_2=2$ units, processing time $t_2=1$, setup $s_2=4$.

The B & B follows the stepwise branching scheme in Section 3.4: (i) Allocation, (ii) backward sequencing, (iii) batching, and then leaf evaluation by backwards-packed scheduling.

4.1.1. Step 1: Allocation screening and lower bound evaluation

For each integer allocation $A_{j,m}$, a quick horizon screening is performed using the minimum-batch workload on each machine (each positive $A_{j,m}$ implies at least one setup). Allocations that violate the due date horizon are discarded. For the remaining allocations, the LB(A) is computed using **Equation 13**. Since the incumbent is initialized as $UB=+\infty$, LB(A)-based dominance pruning is applied only after a feasible incumbent has been obtained. In this example, as shown later in **Step 5**, the detailed evaluation of allocation A6 produces a feasible solution with $F^a=20$, which establishes UB = 20. **Table 2** and **Table 3** summarize the allocation nodes and the post-incumbent pruning decisions for Example 1.

Tables 2 and **3** report the allocation-stage nodes explored by the B & B. **Table 2** lists the integer allocation matrices $A_{1,1}, A_{1,2}, A_{2,1}$, and $A_{2,2}$, while **Table 3** reports the horizon-feasibility screening result, the LB(A), and the pruning or continuation decision after the incumbent (UB = 20) becomes available. Each row represents one candidate allocation of job quantities to the two machines. If "horizon feasible?" is no, the allocation is discarded immediately because even the minimum workload, including at least one setup for each positive allocation, exceeds the due date horizon on at least one machine. As an example, consider Allocation ID A6, where $(A_{1,1}, A_{1,2}, A_{2,1}, A_{2,2})=(1,2,2,0)$: this means Job 1 is split as 1 unit on Machine 1 and 2 units on Machine 2, while Job 2 is fully assigned to Machine 1 (2 units) and none to Machine 2; the row shows "horizon feasible? = yes" and LB(A) = 14, so because LB(A) < UB = 20, allocation A6 is not pruned and is carried forward to the next branching steps.

Table 2. Allocation nodes (Example 1): Integer allocation matrix $A_{j,m}$

Allocation ID	$A_{1,1}$	$A_{1,2}$	$A_{2,1}$	$A_{2,2}$
A1	0	3	0	2
A2	0	3	1	1
A3	0	3	2	0
A4	1	2	0	2
A5	1	2	1	1
A6	1	2	2	0
A7	2	1	0	2
A8	2	1	1	1
A9	2	1	2	0
A10	3	0	0	2
A11	3	0	1	1
A12	3	0	2	0

To provide a clearer visual representation of the B & B search process, **Figures 2–4** are divided into three complementary parts. **Figure 2** presents the allocation-level branching, horizon-feasibility screening, and post-incumbent LB-based pruning. **Figure 3** expands the retained allocation branch A6 and shows how this branch proceeds to the downstream sequencing and batching decision layers. **Figure 4** then illustrates the downstream

branching and optimal solution identification under A6, including the selection of the best feasible branch, backwards-packed scheduling, objective evaluation, and incumbent update.

Table 3. Allocation screening and post-incumbent pruning decisions (Example 1)

Allocation ID	Horizon feasible?	LB(A)	Decision after UB = 20
A1	No	22	Prune (infeasible)
A2	Yes	20	Prune (LB ≥ UB)
A3	Yes	22	Prune (LB ≥ UB)
A4	Yes	14	Continue
A5	Yes	12	Continue
A6	Yes	14	Continued (detailed)
A7	Yes	14	Continue
A8	Yes	12	Continue
A9	Yes	14	Continue
A10	Yes	22	Prune (LB ≥ UB)
A11	Yes	20	Prune (LB ≥ UB)
A12	No	22	Prune (infeasible)

Abbreviations: LB: Lower bound; LB(A): Allocation-based lower bound; UB: Upper bound.

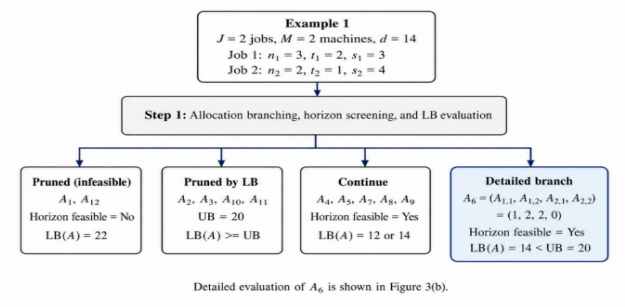

Figure 2. Allocation branching and post-incumbent pruning process of the proposed B & B algorithm for the numerical example
Abbreviations: B & B: Branch-and-Bound; LB: Lower bound; LB(A): Allocation-based lower bound; UB: Upper bound.

The LB-based pruning decisions shown in **Figure 2** are applied after the incumbent $UB = 20$ is obtained from the detailed evaluation of allocation A6.

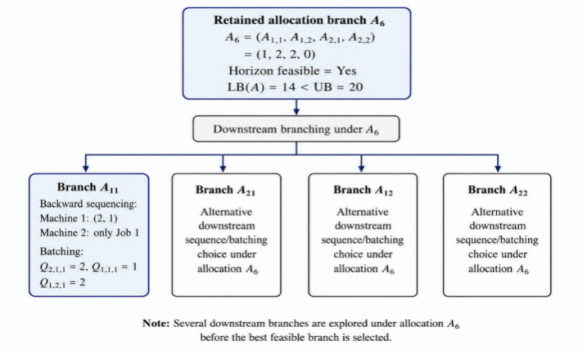

Figure 3. Detailed view of the retained allocation branch A6 in the proposed B & B algorithm
Abbreviations: B & B: Branch-and-Bound; LB(A): Allocation-based lower bound; UB: Upper bound.

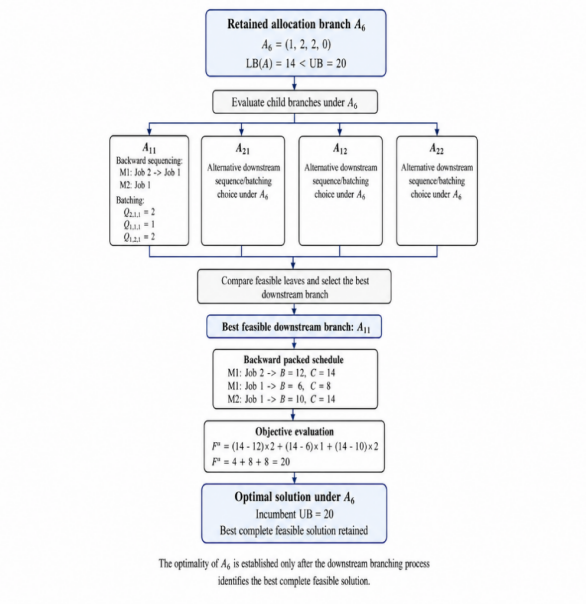

Figure 4. Downstream branching and optimal solution identification under the retained allocation branch A6
Abbreviations: LB(A): Allocation-based lower bound; UB: Upper bound.

4.1.2. Step 2 (backward sequencing)

Under A6, Machine 2 processes only Job 1, so no sequencing decision is needed. Machine 1 processes two job blocks (Job 1 and Job 2). The best backward order is $\pi_m = (2, 1)$, meaning Job 2 is scheduled closest to the due date and Job 1 is scheduled earlier.

4.1.3. Step 3 (batching)

The optimal batching uses one batch per positive allocation (no splitting): on Machine 1, $Q_{2,1,1} = 2$ and $Q_{1,1,1} = 1$; on Machine 2, $Q_{1,2,1} = 2$.

4.1.4. Step 4 (backwards-packed schedule)

A backwards-packed schedule is constructed by anchoring the rightmost batch completion at $d = 14$ on each machine, then placing earlier batches backwards without idle time and with setup performed immediately before processing. The resulting schedule is shown in **Table 4**.

Table 4. Backwards-packed schedule for the optimal branch (A6, Example 1)

Machine	Job	Batch (right → left)	$B_{j,m,i}$	$C_{j,m,i}$
1	2	$Q_{2,1,1} = 2$ (closest to d)	12	14
1	1	$Q_{1,1,1} = 1$	6	8
2	1	$Q_{1,2,1} = 2$ (closest to d)	10	14

4.1.5. Step 5 (objective evaluation)

Using the batch contribution definition, the objective value is computed as follows. On Machine 1: (14 − 12) × 2 = 4 for Job 2 and (14 − 6) × 1 = 8 for Job 1, giving 12. On Machine 2: (14 − 10) × 2 = 8. Therefore, the complete schedule obtained from Allocation A6 gives $F^a = 12 + 8 = 20$. Since the incumbent is initialized to UB = +∞ and is updated whenever a better feasible complete solution is found, the value obtained from A6 improves the incumbent, setting UB = 20. The Gantt chart for this solution is shown in **Figure 5**.

After evaluating feasible allocations, the best feasible objective value obtained for each allocation is summarized in **Table 5**.

Table 5 summarizes, for each allocation ID that passes the feasibility screening, the best feasible total actual flow time F^a obtained after completing the remaining B & B decisions (backward sequencing and integer batching) and evaluating the resulting backward packed schedule. In other words, each value in the table is the minimum F^a achievable within that allocation branch. The results indicate that the global optimum is $F^{a^*} = 20$, achieved by Allocation A6, and the same optimal value is also obtained by A7, reflecting symmetry between identical machines.

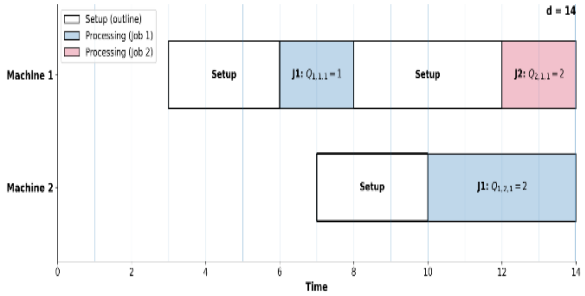

Figure 5. Gantt chart of Allocation ID A6, where $Q_{2,1,1} = 2$, and $Q_{1,1,1} = 1$ are scheduled on Machine 1, and $Q_{1,2,1} = 2$ is scheduled on Machine 2

Table 5. Best feasible F^a per allocation (Example 1)

Allocation ID	Best feasible F^a
A2	29
A3	21
A4	26
A5	24
A6	20
A7	20
A8	24
A9	26
A10	21
A11	29

4.2. Computational experiments and scalability analysis

The computational experiments evaluate the scalability of the proposed B & B algorithm beyond the illustrative example in Section 4.1. 10 controlled test instances were generated by varying the numbers of machines and jobs in small-to-medium exact-test settings, representing situations where multiple jobs compete for limited parallel resources. The instances were designed with small-to-medium demands, controlled processing and setup times, and feasible common due dates so that the exact B & B algorithm could verify global optimality while still capturing the effects of allocation, sequencing, and batching decisions. The analysis focuses on the growth of the allocation search space, feasibility screening, LB pruning, and runtime as the problem size increases.

All experiments were conducted using a custom implementation of the proposed B & B algorithm under identical computational conditions. For each instance, the algorithm was executed until all allocation nodes were either explored or fathomed by infeasibility or dominance; therefore, all reported solutions are certified global optima. **Tables 6** and **7** summarize the scalability results for cases C1–C10 by reporting the allocation search-space size and the computational effort required to prove optimality. The experiments use controlled small-to-medium instances because the number of allocation nodes grows combinatorially as $\prod_{j=1}^{J}\binom{n_j+M-1}{M-1}$. Accordingly, job demands were kept small while M and J were varied to evaluate the behavior of the exact B & B algorithm under tractable exact-test conditions.

Tables 6 and **7** summarize the allocation search space and the computational effort of the exact B & B algorithm. The total number of allocation nodes increases with M, J, and job demand, with the largest tested instance containing 10,240 nodes. Horizon-feasibility screening eliminates allocations whose minimum workload exceeds the common due date horizon, while the $LB(A)$ further prunes selected feasible nodes after an incumbent solution is obtained. Although LB pruning is not dominant in all cases, it helps reduce the search space when the incumbent is sufficiently strong. Overall, the exact B & B algorithm successfully verifies the global optimum for all tested small-to-medium instances, with all runtimes remaining below five seconds.

Table 6. Problem size of the allocation search space

Case	M	J	Total allocation nodes
C1	2	4	36
C2	2	5	144
C3	3	4	162
C4	3	5	972
C5	3	6	2,916
C6	4	4	640
C7	4	5	2,560
C8	4	6	10,240
C9	5	4	1,875
C10	5	5	9,375

5. Discussion

5.1. Key findings and overview

The results in **Tables 2–5** show that the proposed B & B procedure is able to identify the global optimum $F^{a^*}= 20$ for Example 1 and to clearly explain how this value is obtained through allocation screening, backward sequencing, batching decisions, and backwards-packed scheduling. Among all feasible allocations, Allocation A6 (and symmetrically A7) yields the minimum total actual flow time. The optimal solution is characterized by a balanced allocation across machines, a backward job order that places the most critical processing near the common due date, and a single batch for each positive allocation. **Tables 2** and **3** show the allocation

Table 7. B & B performance and outcomes

Case	Horizon-feasible nodes	Pruned by LB (%)	Pruned infeasible	Optimal F^a	Runtime (s)
C1	36	0	0	30	0.0085
C2	144	0	0	76	0.7884
C3	120	0	42	13	0.0451
C4	774	0	198	26	0.3624
C5	2,403	0	513	61	1.3102
C6	384	34.4	256	29	0.0311
C7	1,344	0	1,216	24	0.2184
C8	5,928	0	4,312	26	1.5343
C9	420	92.1	1,455	19	0.0667
C10	2,940	36.7	6,435	16	0.4328

Abbreviations: B & B: Branch-and-Bound; LB: Lower bound.

screening and pruning structure, while **Table 5** confirms that no feasible allocation produces a lower F^a than A6 and A7. **Table 4** illustrates the detailed schedule structure that achieves this optimum.

5.2. Mechanism of the objective under backward JIT scheduling

The solution behavior follows directly from the objective function, since for a given batch size $Q_{j,m,i}$, the contribution to F^a depends entirely on the processing start time $B_{j,m,i}$. The later a batch starts, the smaller its contribution to the total actual flow time. Therefore, schedules that place processing as close as possible to the common due date are preferred, which is precisely the logic of backward JIT scheduling. **Figure 6** illustrates this behavior: panel (A) shows backwards-packed scheduling, where batches are positioned as late as feasible without unnecessary idle time, while panel (B) highlights the setup–splitting trade-off, in which additional splitting introduces extra setups and may force some processing to start earlier, thereby increasing F^a.

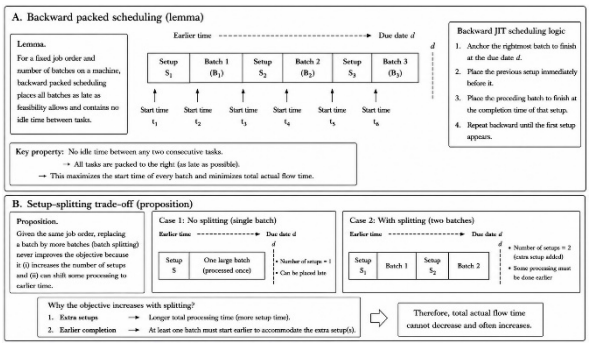

Figure 6. Illustration of backwards-packed scheduling and the setup–splitting trade-off

Figure 6 shows that backwards-packed scheduling places batches as late as possible before the common due date, thereby increasing processing start times and reducing their contribution to total actual flow time. It also highlights the trade-off of batch splitting: although splitting may improve scheduling flexibility, it introduces additional setup time that consumes the available time window and may force some batches to start earlier. This explains why backwards-packed schedules appear in the optimal B & B leaves; any unnecessary idle time would reduce $B_{j,m,i}$, increase $(d - B_{j,m,i})$, and worsen the objective value.

5.2.1 Lemma 1

For any fixed combination of Allocation A, backward job order π, and batch sizes Q, a backwards-packed schedule (with no unnecessary idle time and setup performed immediately before processing) yields the minimum possible value of F^a among all feasible schedules consistent with $\left(A_{j,m}, \pi_m, Q_{j,m,i}\right)$.

5.2.2. Proof

Given $\left(A_{j,m}, \pi_m, Q_{j,m,i}\right)$, inserting idle time before any batch shifts that batch and all batches to its left earlier in time, reducing $B_{j,m,i}$ while leaving $Q_{j,m,i}$ unchanged. Since $\left(d - B_{j,m,i}\right)Q_{j,m,i}$ increases when $B_{j,m,i}$ decreases, any idle time strictly increases F^a. Therefore, the backwards-packed schedule, which maximizes all feasible batch start times, is optimal for leaf evaluation.

5.2.3. Setup–splitting trade-off and the role of batching

The results also clarify why batch splitting is not selected in the optimal solution of Example 1. Each additional batch increases the number of setups $N_{j,m}$, which directly affects the horizon-feasibility through the constraint $\sum_{j=1}^{J}\left(t_j A_{j,m} + s_j N_{j,m}\right) \le d$. Additional setup time consumes capacity near the due date, forcing some processing to be shifted earlier. This shift reduces batch start times $B_{j,m,i}$ and increases the corresponding contribution to F^a. As a result, although splitting can sometimes improve flexibility, it becomes counterproductive when setup times are non-negligible relative to processing times or when the due date horizon is tight.

In Example 1, setup times are substantial for both jobs ($s_1 = 3$ and $s_2 = 4$), and the horizon is limited to $d = 14$. Under these conditions, introducing additional batches would add setup overhead that outweighs any potential benefit of finer batch positioning, leading to a higher total actual flow time.

5.2.4. Proposition 1

When setup times are relatively large compared to unit processing times, or when the common due date is tight, solutions with a minimal number of batches (often one batch per positive allocation) tend to minimize F^a.

5.2.5. Proof

Each additional batch introduces a fixed setup that reduces the time available for processing as the due date approaches. This loss must be compensated for by shifting some processing earlier, which increases $\left(d - B_{j,m,i}\right)$ for the affected units. Consequently, in a setup-dominant regime, minimizing the number of batches reduces early starts and leads to lower F^a.

5.2.6. Allocation A6 (and A7) becomes optimal

Table 5 shows that Allocation A6 achieves the minimum value $F^a = 20$, while several other allocations produce strictly larger values. The key reason is that A6 balances processing loads across the two machines without introducing excessive setup overhead. On Machine 1, both jobs are processed, but the backward order places Job 2 closest to the due date; on Machine 2, Job 1 alone occupies the final segment of the horizon. This configuration allows most units to be processed close to the due date while keeping setup counts to a minimum. Allocation A7 yields the same objective value because the machines are identical. Swapping the roles of Machines 1 and 2 produces a symmetric schedule with identical batch start times relative to the due date. This symmetry explains the existence of multiple optimal solutions and highlights the importance of tie-breaking rules in practical implementations, for example, based on schedule stability or secondary criteria.

5.3. Bounding, pruning, and computational implications

From a computational perspective, **Table 3** illustrates the basic pruning logic of the proposed B & B procedure. Horizon screening first discards allocations that cannot fit within the due date window, while the LB(A) is used after an incumbent solution is obtained to eliminate branches that cannot improve the current best solution. Although the illustrative example is small, the same logic applies to the computational experiments, where the exact B & B algorithm evaluates controlled small-to-medium instances with allocation search spaces of up to 10,240 nodes.

The computational results also position the proposed B & B algorithm relative to existing scheduling studies. A direct numerical comparison is difficult because prior studies typically differ in their machine environments, due date structures, objective functions, and benchmark datasets. In contrast, this study integrates allocation, integer batching, sequencing, job-dependent setups, and backward common due date scheduling under the total actual flow time objective. The proposed B & B algorithm, therefore, serves primarily as an exact benchmark method: it certifies global optimality for all tested instances and completes all runs in under five seconds, providing reliable reference solutions for future heuristic or hybrid approaches on larger-scale settings.

5.4. Managerial insights

Several practical insights can be drawn from the example. First, backward JIT scheduling is highly effective for reducing total actual flow time because it naturally minimizes early starts and WIP. Second, batch splitting should not be applied indiscriminately; when setup times are significant, minimizing the number of batches can be more beneficial than increasing flexibility. Third, allocation decisions must account not only for processing balance but also for setup placement and proximity to the due date. Finally, in systems with identical parallel machines, multiple optimal solutions may exist, and additional operational criteria may be needed to select among them.

The computational results from experiments across increasing problem sizes indicate that exact optimization is most suitable for small to medium-sized planning problems and should primarily be used as a benchmarking and validation tool. As problem size increases, the rapid growth in computation time suggests that fast and robust heuristics become more valuable than guaranteed optimality. Managers should therefore align decision-support methods with system scale: using optimal models to understand structural properties and calibrate rules, while relying on efficient heuristic approaches for large-scale, time-critical scheduling decisions.

6. Conclusion

This study investigated a discrete multi-job batch scheduling problem on identical parallel machines under a common due date, with the objective of minimizing total actual flow time within a backward JIT scheduling framework. The model integrates demand allocation, integer batching, per-machine sequencing, and job-dependent setup positioning. An exact B & B algorithm was developed using stepwise branching over allocation, sequencing, and batch-size composition decisions, with feasible leaf nodes evaluated through backwards-packed scheduling.

The illustrative example shows that optimality is determined not only by workload balance alone, but also by setup consumption, batch formation, and backward schedule positioning. The computational results further show that the proposed B & B algorithm certifies global optimality for all controlled small-to-medium benchmark instances, with allocation search spaces of up to 10,240 nodes and runtime below five seconds. These findings confirm the role of the proposed B & B as an exact benchmark method for validating future heuristic, hybrid, or learning-based algorithms.

The main limitation of this study is that the exact B & B algorithm remains computationally demanding for larger industrial-scale instances because the search space grows combinatorially with the number of jobs, machines, and feasible batch-size compositions. Computational analysis is also based on constructed benchmark instances rather than real industrial data. Future research may develop heuristic, metaheuristic, hybrid, or learning-based algorithms for larger-scale problems using the optimal solutions generated by the proposed B & B as benchmark references.[31,32,44,45] Further extensions may consider heterogeneous machine settings,[31,33,50] sequence-dependent setups, resource constraints, and machine availability,[26,32,50] as well as uncertainty, energy-related objectives, carbon-emission measures, and WIP-related sustainability indicators.[4–7,29,43] Finally, because batching decisions are closely related to downstream logistics and delivery coordination, future work may extend the model to integrated production–delivery systems under common due date or JIT environments.[14,27]

Acknowledgments

The authors would like to express their sincere gratitude to the Manufacturing Systems Research Group at Universitas Jenderal Achmad Yani and the Advanced Manufacturing Research Group, Faculty of Mechanical Engineering, Universiti Teknologi Malaysia, for their valuable support and academic contribution to this research.

Funding

This research was supported by the Internal Research Funding Scheme of Universitas Jenderal Achmad Yani (UNJANI), Indonesia, as stipulated in Rector's Decree No. Skep/188/Unjani/VI/2025 regarding the determination of internal research funding 2025.

Conflict of interest

The authors declare that there are no conflicts of interest with any person, institution, or organization related to the content of this manuscript.

Author contributions

Conceptualization: Rinto Yusriski
Formal analysis: All authors
Investigation: Mohd Azlan Suhaimi
Methodology: All authors
Software: Rinto Yusriski, Andri Rachmat Kumalasian Nasution
Validation: Andri Rachmat Kumalasian Nasution
Writing – original draft: Rinto Yusriski, Andri Rachmat Kumalasian Nasution
Writing – review & editing: Rinto Yusriski, Andri Rachmat Kumalasian Nasution

Availability of data

The data that support the findings of this study are available on Zenodo at https://doi.org/10.5281/zenodo.20503546 (record number 20503546). The dataset contains the benchmark instances generated and the computational results used in this study. The dataset does not involve human subjects, personal information, or sensitive content.

AI tools statement

The authors used Grammarly and ChatGPT to improve the manuscript's grammar, clarity, and readability. These tools were used for language editing only and did not generate scientific content, results, interpretations, or conclusions. All authors confirm that AI tools were used only for language editing and that all final decisions and responsibility for the manuscript content remain with the authors.

References

1. Liu L, Yan C-B, Li J. Modeling, analysis, and improvement of batch-discrete manufacturing systems: A systems approach. *IEEE Trans Autom Sci Eng.* 2022;19(3):1567–1585.

   https://doi.org/10.1109/TASE.2021.3127048

2. Shakeri Z, Benfriha K, Varmazyar M, Talhi E, Quenehan A. Production scheduling with multi-robot task allocation in a real industry 4.0 setting. *Sci Rep.* 2025;15(1):1795.

https://doi.org/10.1038/s41598-024-84240-3

3. Muñoz FT, Ulloa-Navarro J. A make-to-order capacitated lot-sizing model with parallel machines, eligibility constraints, extra shifts, and backorders. *Mathematics*. 2025;13(11):1798.

   https://doi.org/10.3390/math13111798

4. Puka R, Skalna I, Duda J, Derlecki T. EPO extension of dispatching rules to minimize effects of time uncertainty in production scheduling. *Appl Sci*. 2025;15(13):7408.

   https://doi.org/10.3390/app15137408

5. Dey BK, Pareek S, Tayyab M, Sarkar B. Autonomation policy to control WIP inventory in a smart production system. *Int J Prod Res*. 2021;59(4):1258–1280.

   https://doi.org/10.1080/00207543.2020.1722325

6. Sarkar B, Mridha B, Pareek S. A sustainable smart multi-type biofuel manufacturing with the optimum energy utilization under flexible production. *J Clean Prod*. 2022;332:129869.

   https://doi.org/10.1016/j.jclepro.2021.129869

7. Sarkar B, Guchhait R, Sarkar M. A sustainable electric vehicle smart production with WIP inventory of outsourced spare parts. *J Ind Inf Integr*. 2026;49:100998.

   https://doi.org/10.1016/j.jii.2025.100998

8. Halim AH, Miyazaki S, Ohta H. Batch-scheduling problems to minimize actual flow times of parts through the shop under JIT environment. *Eur J Oper Res*. 1994;72(3):529–544.

   https://doi.org/10.1016/0377-2217(94)90421-9

9. Hidayat NP, Aribowo W, Halim AH. A single batch processor scheduling model with additional setup times to minimize total actual flowtime of parts of multiple items. *J Phys Conf Ser*. 2021;1858(1):012019.

   https://doi.org/10.1088/1742-6596/1858/1/012019

10. Suryadhini PP, Sukoyo, Suprayogi, Halim AH. A batch scheduling model for a three-stage flow shop with job and batch processors considering a sampling inspection to minimize expected total actual flow time. *J Ind Eng Manag*. 2021;14(3):520–537.

    https://doi.org/10.3926/jiem.3438

11. Yusriski R, Sukoyo, Samadhi TMA, Halim AH. Integer batch scheduling problems for a single machine with simultaneous effect of learning and forgetting to minimize total actual flow time. *Int J Ind Eng Comput*. 2015;6(3):365–378.

    https://doi.org/10.5267/j.ijiec.2015.2.005

12. Yusriski R, Sukoyo, Samadhi TMA, Halim AH. An integer batch scheduling model considering learning, forgetting, and deterioration effects for a single machine to minimize total inventory holding cost. *IOP Conf Ser Mater Sci Eng*. 2018;319:012038.

    https://doi.org/10.1088/1757-899X/319/1/012038

13. Yusriski R, Astuti B, Biksono D, Wardani TA. A single machine multi-job integer batch scheduling problem with multi due date to minimize total actual flow time. *Decis Sci Lett*. 2021;10(3):231–240.

    https://doi.org/10.5267/j.dsl.2021.4.002

14. Rahmawati S, Masruroh NA, Rifai AP. Integrated production and delivery batching for simultaneous multi-job scheduling in multi-factory environments: Modeling, linearization, and optimization. *Prod Eng*. 2026;20:4.

    https://doi.org/10.1007/s11740-025-01398-z

15. Mao Z, Cao T, Fu E, Fang K, Huang D. An adaptive hybrid neighbourhood search algorithm for order acceptance and batch scheduling problem in additive manufacturing. *Int J Prod Res*. 2025;1–25.

    https://doi.org/10.1080/00207543.2025.2516189

16. Yusriski R. Application of the overlapping method to the flow shop scheduling problem to minimize the makespan. *Int J Technol Energy*. 2025;1(3):107–122.

    https://doi.org/10.71364/ijte.v1i3.14

17. Badri SA, Ghazanfari M, Makui A. An integrated model for product mix problem and scheduling considering overlapped operations. *Decis Sci Lett*. 2014;3(4):523–534.

    https://doi.org/10.5267/j.dsl.2014.5.006

18. Kurniawan D, Yusriski R, Isnaini MM, Ma'ruf A, Halim AH. A flow shop batch scheduling model with pre-processing and time-changing effects to minimize total actual flow time. *J Ind Eng Manag*. 2024;17(2):542–561.

    https://doi.org/10.3926/jiem.7134

19. Sibarani AA, Setyaningrum DT, Waluyu S. Multi-item flow shop batch scheduling with two stages dedicated machines to minimize total actual flow time. *Int J Integr Eng*. 2025;17(1):140–153.

    https://doi.org/10.30880/ijie.2025.17.01.012

20. Sibarani AA, Prabowoputra DM, Melita P. Modeling batch scheduling in a three-stage flow shop with multi-item on batch-job-batch processors to minimize total actual flow time. *Ind Eng Manag Syst*. 2025;24(2):188–201.

    https://doi.org/10.7232/iems.2025.24.2.188

21. Maulidya R, Suprayogi, Wangsaputra R, Halim

AH. Batch scheduling of unique and common components for a three-stage hybrid flow shop processing different product types with multiple due dates to minimize total actual flow time. *In: Intelligent Engineering and Management for Industry 4.0*. Springer; 2022:13–24.

https://doi.org/10.1007/978-3-030-94683-8_2

22. Suryadhini PP, Sukoyo, Suprayogi. A batch scheduling model for a three-stage flowshop with batch processor and heterogeneous job processor to minimize total actual flowtime. *In: Intelligent Engineering and Management for Industry 4.0*. Springer; 2022:1–12.

https://doi.org/10.1007/978-3-030-94683-8_1

23. Yusriski R, Nasution ARK, Lukas L, Wijayanti L, Octaviani S. A two-machine flow shop batch scheduling model to minimize total actual flow time. *J Tek Ind*. 2023;25(2):179–194.

https://doi.org/10.9744/jti.25.2.179-194

24. Medeiros JMF, Subramanian A, Queiroga E. Population-based iterated local search for batch scheduling on parallel machines with incompatible job families, release dates, and tardiness penalties. *Optim Lett*. 2025;19(1):193–210.

https://doi.org/10.1007/s11590-024-02116-x

25. Yang H, Du Y, Li Y, Qian W, Hu B. A heuristic mutation based genetic algorithm for fast parallel scheduling of steel cold rolling. *Chin J Mech Eng*. 2025;38:100.

https://doi.org/10.1186/s10033-025-01271-1

26. Zhang L, Li S, Geng Z, Chen R, Xu J. Multiple parallel-batch machines scheduling with additive resource assignment and machine available times. *Discrete Appl Math*. 2026;379:675–684.

https://doi.org/10.1016/j.dam.2025.09.035

27. Mazdeh MM, Heydari M, Karamouzian A. An integrated model of scheduling, batch delivery and supplier selection in a make-to-order manufacturing system. *Decis Sci Lett*. 2016;5(2):189–200.

https://doi.org/10.5267/j.dsl.2015.12.005

28. Zahedi Z, Salim A, Yusriski R, Haris H. Optimization of an integrated batch production and maintenance scheduling on flow shop with two machines. *Int J Ind Eng Comput*. 2019;10(2):225–238.

https://doi.org/10.5267/j.ijiec.2018.7.001

29. Chen Y, Xu L, Rauf M, Li P, Mumtaz J. Multi-objective artificial bee colony algorithm for energy-efficient scheduling of unrelated parallel batch processing machines with flexible preventive maintenance. *Int J Ind Eng Comput*. 2025;16(3):619–640.

https://doi.org/10.5267/j.ijiec.2025.4.008

30. Fang W, Wang Z, Luo D. Research on integrated optimization of order allocation and lot-sizing sequencing for mixed-model parallel assembly lines using improved intelligent optimization algorithm. *Int J Ind Eng Comput*. 2026;17(1):31–50.

https://doi.org/10.5267/j.ijiec.2025.12.004

31. ElWakil M, Lersteau C, Shen W, Sabry I. Optimising batch scheduling on non-identical parallel machines: Lower bounds, MIP, branch-and-price, and heuristics. *Int J Prod Res*. 2025;63(24):9722–9747.

https://doi.org/10.1080/00207543.2025.2524521

32. Santos MEP, Silva YLTV, Araújo MCB de. An adaptive local search for large-scale parallel machine scheduling in textile production with release dates and sequence-dependent setup times. *Int J Ind Eng Comput*. 2025;16(3):693–708.

https://doi.org/10.5267/j.ijiec.2025.4.004

33. Ou J, Li W. Scheduling mixed batch machines with inclusive processing set restrictions and non-identical capacities. *Eur J Oper Res*. 2026;328(2):407–414.

https://doi.org/10.1016/j.ejor.2025.07.012

34. Sun T, Wang J-Q, Fan G-Q, Liu Z. Generalized parallel batch scheduling: Batch processing time non-decreasing in job width and thickness. *Omega*. 2026;103558.

https://doi.org/10.1016/j.omega.2026.103558

35. Fan G-Q, Wang J-Q, Liu Z. A scheduling game on parallel batch machines with setup cost. *J Sched*. 2026;29:143–156.

https://doi.org/10.1007/s10951-025-00863-y

36. Huang Y, Zhou B. A Tabu-Bi-label hybridized branch and price algorithm for just-in-time material handling scheduling problems of mixed-model assembly lines with electric vehicle recharging requirements. *J Heuristics*. 2025;31(3):27.

https://doi.org/10.1007/s10732-025-09563-4

37. Ozturkoglu O. Identical parallel machine scheduling with nonlinear deterioration and multiple rate modifying activities. *Int J Optim Control Theor Appl*. 2017;7(2):167–176.

https://doi.org/10.11121/ijocta.01.2017.00328

38. Demir HI, Canpolat O. Integrated process planning, WSPT scheduling and WSLK due-date assignment using genetic algorithms and evolutionary strategies. *Int J Optim Control Theor Appl*. 2018;8(1):73–83.

https://doi.org/10.11121/ijocta.01.2018.00412

39. Karakaş E, Özpalamutçu H. A genetic algorithm

for fuzzy order acceptance and scheduling problem. *Int J Optim Control Theor Appl.* 2019;9(2):186–196.

https://doi.org/10.11121/ijocta.01.2019.00711

40. Arık OA. Optimal policies for minimizing total job completion times and deviations from common due dates in unrelated parallel machine scheduling. *OPSEARCH.* 2024;61(3):1654–1680.

https://doi.org/10.1007/s12597-024-00750-8

41. Mor B, Shapira D, Sonn N. Minimizing the total weighted earliness–tardiness about a non-restrictive common due date with optional job rejection on single and multiple machines. *Comput Appl Math.* 2026;45(5):217.

https://doi.org/10.1007/s40314-025-03586-0

42. Yang Y, Ding H, He W. Online budget allocation maximization problem on two uniform machines with a common due date. *In: International Computing and Combinatorics Conference.* Vol 15983. Springer; 2026:346–357.

https://doi.org/10.1007/978-981-95-0215-8_26

43. Kucukkoc I. Scheduling of distributed additive manufacturing machines considering carbon emissions. *Int J Optim Control Theor Appl.* 2024;14(1):20–31.

https://doi.org/10.11121/ijocta.1444

44. Huertas JA, Hentenryck PV. Parallel Batch Scheduling with Incompatible Job Families via Constraint Programming. *IEEE Trans Semicond Manufact.* 2025;38(4):901-916.

https://doi.org/10.1109/TSM.2025.3601510

45. Rolim GA, Tomazella CP, Nagano MS. On the integration of reinforcement learning and simulated annealing for the parallel batch scheduling problem with setups. *Eur J Oper Res.* 2025;326(2):220–233.

https://doi.org/10.1016/j.ejor.2025.04.042

46. Sun T, Wang JQ, Fan GQ, Liu Z. Submodular batch scheduling on parallel machines. *Nav Res Logist.* 2025;72(2):242–259.

https://doi.org/10.1002/nav.22221

47. Shahvari O, Logendran R, Tavana M. An efficient model-based branch-and-price algorithm for unrelated-parallel machine batching and scheduling problems. *J Sched.* 2022;25(5):589–621.

https://doi.org/10.1007/s10951-022-00729-7

48. Li Y, Côté JF, Callegari-Coelho L, Wu P. Novel formulations and logic-based benders decomposition for the integrated parallel machine scheduling and location problem. *INFORMS J Comput.* 2022;34(2):1048–1069.

https://doi.org/10.1287/ijoc.2021.1113

49. Chen J. Branch-and-price algorithms for parallel machine scheduling with machine usage costs. Doctoral dissertation. Université Gustave Eiffel; Université de Technologie de Hefei; 2025. Accessed July, 8, 2026. https://theses.hal.science/tel-05473231v1/document

50. Yusriski R, Nasution ARK, Isnaini MM, Halim AH. A batch scheduling model on unrelated parallel machines with resource constraints and sequence-dependent setup time to minimize total actual flow time. *J Ind Eng Manag.* 2026;19(1):58–81.

https://doi.org/10.3926/jiem.8710

Rinto Yusriski *is a faculty member in the Department of Industrial Engineering, Faculty of Manufacturing Technology, Universitas Jenderal Achmad Yani (UNJANI), Bandung, Indonesia. His research focuses on production planning and scheduling, integer batch scheduling, and exact/heuristic optimization methods for manufacturing systems. He is particularly interested in integrated decision-making that links allocation, batching, and sequencing under practical constraints such as setup times and due-date-driven production environments. His recent work includes the development of mathematical models and algorithmic procedures to support just-in-time-oriented performance measures, such as total actual flow time.*
http://orcid.org/0000-0001-8933-9518

Andri Rachmat Kumalasian Nasution *is affiliated with the Department of Industrial Engineering, Faculty of Manufacturing Technology, Universitas Jenderal Achmad Yani (UNJANI), Bandung, Indonesia. His interests include operations research, mathematical modeling for scheduling, and algorithm development, including exact methods such as Branch-and-Bound and computational evaluation of optimization procedures. His work emphasizes discrete manufacturing problems involving parallel machines, batch-size decisions, and setup-related constraints, with the goal of producing decision-support approaches that are both rigorous and applicable to real production settings.*
http://orcid.org/0000-0001-5753-3578

Mohd Azlan Suhaimi *is a member of the Advanced Manufacturing Research Group, Faculty of Mechanical Engineering, Universiti Teknologi Malaysia (UTM), Malaysia. His research covers advanced manufacturing systems, manufacturing operations optimization, and decision-support methods that bridge engineering applications with operations research. He is interested in developing and vali-*

dating optimization-based frameworks for complex production environments, including batch processing and parallel-machine scheduling under industrially relevant constraints.
http://orcid.org/0000-0001-2115-2030

An International Journal of Optimization and Control: Theories & Applications (https://accscience.com/journal/ijocta)