Proceedings of the International Conference on Industrial Engineering and Operations Management

Publisher: IEOM Society International, USA DOI: 10.46254/AN16.20260031
Published: June 23, 2026

An Adaptive Large Neighborhood Search for Shift-Dependent Scheduling on Unrelated Parallel Machines with Sequence-and-Machine-Dependent Setups

Demiray Demir and Milad Elyasi
Industrial Engineering Department
Özyeğin University
Istanbul, Turkey
demiray.demir@ozu.edu.tr, milad.elyasi@ozyegin.edu.tr

Abstract

This paper studies an unrelated parallel machine scheduling problem arising in heterogeneous manufacturing systems. The production environment features machine-dependent processing times, sequence-dependent setup times, and workforce-driven productivity differences modeled through two non-cyclic shifts with distinct processing rates. Jobs must be assigned and sequenced on eligible machines, and the objective is to minimize total tardiness. The problem is computationally challenging due to the strong coupling between assignment, sequencing, and timing decisions induced by setup effects and shift-dependent processing speeds. A Mixed-Integer Linear Programming (MILP) formulation is developed to obtain optimal solutions for small instances. For large-scale instances, we propose an Adaptive Large Neighborhood Search (ALNS) algorithm that combines a constructive initial heuristic with multiple destruction and repair operators. Computational experiments show that the proposed ALNS consistently outperforms state-of-the-art MILP solvers and benchmark metaheuristics.

Keywords

Unrelated parallel machine scheduling, Sequence-and-machine-dependent setups, Shift-dependent productivity, Tardiness minimization, Adaptive large neighborhood search

1. Introduction

Many high-volume manufacturing systems operate under tight delivery commitments, increasing product variety, and significant heterogeneity in production resources. In such environments, short-term production scheduling plays a critical role in ensuring on-time delivery and maintaining service levels. Jobs must be assigned and sequenced on any of the multiple parallel machines that differ in processing capabilities and speeds, while adjusting for setup requirements and workforce-related productivity differences. As delays propagate quickly across downstream operations, minimizing total job tardiness has become a primary scheduling objective in practice (Xu et al. 2016). A common modeling framework for capturing machine heterogeneity is the Unrelated Parallel Machine (UPM) scheduling problem, where processing times depend on both the job and the machine to which it is assigned. However, many real-world production systems exhibit additional features that are not fully captured by classical UPM formulations. In particular, sequence-dependent setup times are prevalent when switching between jobs with different characteristics, directly coupling assignment and sequencing decisions (Allahverdi 2015). Another important source of complexity arises from variations in workforce-related productivity in shifts. Production lines often operate in multiple daily shifts that differ in staffing levels, operator experience, and working pace. These differences translate into shift-dependent machine processing rates, which directly affect job completion times. In this study, we consider a planning horizon with two non-cyclic shifts, where processing speeds differ between shifts and do not repeat within

the horizon. Unlike cyclic shift models, this setting requires explicit consideration of when jobs are processed relative to shift boundaries, as delaying a job into a later shift may significantly increase its tardiness.

Motivated by these characteristics, this paper studies an Unrelated Parallel Machine Scheduling Problem with Sequence-Dependent Setups and Non-Cyclic Shifts (UPMS-SDS-NCS). The problem consists of assigning and sequencing a set of jobs on unrelated parallel machines subject to machine eligibility constraints and sequence-dependent setup times. Processing times depend on both the selected machine and the shift during which processing occurs. Each job has a due date, and the objective is to minimize total tardiness across all jobs. This integrated formulation captures key operational trade-offs faced by planners, where assignment, sequencing, and timing decisions must be coordinated under shift-dependent productivity effects. From a computational perspective, the resulting problem is highly challenging. Even the classical UPM problem with sequence-dependent setups and tardiness minimization is strongly NP-hard (Pinedo 2016). The introduction of non-cyclic shift-dependent processing rates further increases complexity by coupling scheduling decisions with time-dependent machine speeds. While a Mixed-Integer Linear Programming (MILP) formulation can model the problem accurately and provide optimal solutions for small instances, exact approaches become impractical for larger problem sizes encountered in realistic settings. To address this challenge, we propose an Adaptive Large Neighborhood Search (ALNS) algorithm tailored to the structure of the problem. The ALNS framework employs multiple destroy and repair operators that modify job-to-machine assignments and sequencing decisions, while explicitly accounting for sequence-dependent setups and shift-dependent processing times. An adaptive mechanism adjusts operator selection based on historical performance, enabling efficient exploration of the large and combinatorial solution space.

We formulate an unrelated parallel machine scheduling problem that integrates sequence-dependent setup times and non-cyclic shift-dependent processing speeds, with the objective of minimizing total tardiness.

1. We develop a specialized ALNS algorithm that effectively addresses the strong coupling between assignment, sequencing, and timing decisions induced by setup and shift effects.
2. We provide extensive computational experiments demonstrating that the proposed ALNS achieves an average improvement of 7.3% over exact solver benchmarks for small and medium instances, and outperforms a state-of-the-art constructive heuristic by approximately 6% on large-scale instances.

The remainder of the paper is organized as follows. Section 1 reviews the related literature. Section 2 provides a detailed problem description. Section 3 presents the MILP formulation. Section 4 describes the ALNS solution methodology. Section 5 reports computational results, and Section 6 concludes the paper and outlines future research directions.

2. Literature Review

Parallel-machine environments constitute a major class of single-stage scheduling problems and are commonly categorized into three fundamental settings. *Identical parallel machines*: all machines are interchangeable, and a job's processing time is independent of the machine to which it is assigned. *Uniform (related) parallel machines:* machines differ only by their constant speeds, so a consistent ordering of machines by capacity can be established; processing times scale according to machine speed. *Unrelated parallel machines:* processing times depend on the specific job–machine pair, meaning that the same machine may process different jobs at different rates, and a job may exhibit substantially different processing requirements across machines.

In its basic form, the UPMSP has been extensively studied in the literature (Vallada and Ruiz 2011; Mecler et al. 2022). However, when the problem is extended to incorporate richer and more realistic constraints—such as complex eligibility structures, sequence-dependent setups (Xie et al. 2025), reconfiguration requirements, workforce/shift effects (Afzalirad and Rezaeian 2016; Chen 2009), or additional operational side constraints—the resulting UPMSP variants become considerably more challenging. For these highly constrained and practically motivated versions, the literature remains comparatively less comprehensive, indicating clear opportunities for further methodological development and empirical investigation (Đurasević and Jakobović 2023). Berthier et al. (2022) study an Unrelated Parallel Machine Scheduling Problem (UPMSP) that incorporates sequence-dependent setup times and machine-eligibility constraints. Their objective is to minimize the makespan, subject to limited resource availability. To solve the problem, they propose a genetic algorithm initialized from randomly generated solutions.

Fang et al. (2022) also study the Unrelated Parallel Machine Scheduling Problem (UPMSP) with sequence-dependent setup times. Their objective based on makespan minimization. They propose a hybrid metaheuristic, LA-ALNS-TS, which integrates an ALNS-based search with Tabu Search. The authors note that pure metaheuristics can suffer from

short-term cycling (revisiting similar solutions), and they suggested to use the tabu mechanism to diversify the search and reduce cycling. This hybrid approach aligns with the success of ALNS in handling complex setup constraints as demonstrated in Xie et al. (2025) and Ropke and Pisinger (2006). Unlike the others, Yanıkoğlu and Yavuz (2022) study the UPMSP under processing-time uncertainty, where the exact durations of jobs may deviate from their nominal values. Their objective is to minimize the worst-case tardiness, providing solutions that remain feasible and effective under adverse realizations of processing times. This contrasts with standard deterministic tardiness minimization approaches found in Logendran et al. (2007) and Li et al. (2021). To achieve this, they formulate a robust counterpart of the scheduling model using Robust Optimization techniques and solve it with an exact Branch-and-Price approach. Uribe et al. (2025) propose a novel mathematical formulation for the UPMSP that integrates machine availability and eligibility constraints within a shift-based operating framework. Unlike traditional approaches, the number of activated shifts is explicitly modeled as a key decision variable. The authors argue that minimizing the completion time of the last shift, rather than the conventional makespan, leads to lower operational costs—such as labor and energy consumption—by preventing the activation of unnecessary shifts. Santoro and Junqueira (2023) address the UPMSP with Machine Availability and Eligibility Constraints by modeling machine availability as a fixed calendar consisting of active and inactive time windows. Unlike approaches that treat shifts as decision variables, their work focuses on the mathematical structure of the problem. In particular, they distinguish between standard preemption and a novel pause constraint, in which jobs may be paused at shift boundaries and resumed later without incurring any penalty. Rather than modifying the objective function, the authors propose lifted reformulations—enhanced MILP models that incorporate valid inequalities derived from problem relaxations—to strengthen lower bounds and more efficiently minimize the standard makespan. Dou et al. (2021) address the Integrated Configuration Design and Scheduling problem in Reconfigurable Manufacturing Systems, considering a dynamic production environment in which the manufacturing system evolves over multiple demand periods. Unlike conventional scheduling problems that assume a fixed shop-floor configuration, their model explicitly accounts for system reconfiguration over time. The problem is formulated as a bi-objective optimization model that simultaneously minimizes total cost—including machine investment and reconfiguration costs—and total tardiness (similar to the weighted tardiness objectives in Lee and Pinedo 1997). To solve this NP-hard problem, the authors develop a Multi-Objective Particle Swarm Optimization algorithm. Ekici et al. (2019) study the UPMSP with machine eligibility constraints and sequence-dependent setup times. In addition, workload balance across machines is addressed by imposing lower and upper bounds on the number of jobs assigned to each machine. The problem objective simultaneously minimizes total tardiness and total earliness. To solve the problem, they discuss several solution approaches and combines them within a hybrid framework. In particular, a Sequential Heuristic Approach (SHA) is proposed, which decomposes the problem into sequential mathematical programming models for job assignment and scheduling. Furthermore, a Tabu Search algorithm is developed and integrated with SHA to enhance solution quality. Building on similar complex industrial constraints, Elyasi et al. (2024) investigate the UPMSP with sequence-and-machine-dependent setups, machine-job compatibility, and workload balancing. They proposed an Imperialist Competitive Algorithm to minimize total earliness and tardiness.

While many studies focus on simplified settings, real-world production environments are often characterized by multiple interacting constraints and sources of heterogeneity. In this paper, we consider a comprehensive UPMSP setting that simultaneously incorporates job-specific release times, sequence- and machine-dependent setup times, workload balance constraints, machine reconfiguration decisions, and non-cyclic shift structures. The integration of these features significantly increases the modeling realism of the problem, bringing it closer to practical manufacturing systems where machines operate under varying configurations, labor availability, and operational policies. The resulting problem constitutes a highly complex NP-hard optimization problem, for which exact solution approaches become computationally prohibitive even for moderate-sized instances. To effectively address this complexity, we propose an ALNS framework enhanced with a diverse set of destroy and repair operators, combined with a Simulated Annealing–based acceptance criterion. This hybrid metaheuristic is designed to efficiently explore the solution space, balance intensification and diversification, and produce high-quality solutions within reasonable computational times.

3. Problem Definition

This paper studies a UPMS-SDS-NCS, motivated by high-volume manufacturing lines that operate with heterogeneous equipment, configuration-dependent processing capabilities, and within-horizon workforce variability. We are given a set of jobs $\mathcal{I} = \{1, \dots, \bar{I}\}$ to be processed on a set of unrelated parallel machines $\mathcal{L} = \{1, \dots, \bar{L}\}$. Each machine $l \in \mathcal{L}$ can operate under one of a finite set of configurations $\mathcal{C} = \{1,2\}$, capturing retooling, fixture changes, program selection, or operator-dependent operating modes. The processing time of job i depends on both the machine and its configuration, and is denoted by P_{ilc}. Not all job–machine-configuration combinations are feasible. The

feasibility is captured by an eligibility indicator $L_{ilc} \in \{0,1\}$, and we define the set of eligible triples as $\mathcal{E} = \{(i,l,c): L_{ilc} = 1\}$. Jobs are subject to release times R_i and due dates D_i. The performance measure is tardiness-based: completing jobs after their due dates degrades service levels and increases downstream disruption. Consequently, the objective is to minimize total tardiness $\sum_{i\in\mathcal{I}} u_i$, where $u_i = \max\{0, f_i - D_i\}$ and f_i denote the completion time of the job i.

In addition to processing times, the production system incurs two types of changeover losses on each machine: (i) *sequence-dependent* setup times T_{ijl} when job j follows job i on machine l, and (ii) a *configuration-change penalty* when consecutive jobs require different configurations. The latter is modeled by a binary indicator w_{ijl} and a penalty parameter CFG that is charged whenever a configuration change occurs from i to j on machine l. We explicitly capture shift-driven productivity variations through *non-cyclic* within-horizon shift segments. Let H_1 and H_2 denote the boundaries defining three consecutive segments over the planning horizon, indexed by $\mathcal{D} = \{0,1,2\}$. Under the *start-time based* (non-preemptive) rule adopted here, the effective processing duration of a job is determined solely by the segment in which it *starts*. This is captured via the rate multiplier $\Gamma(d) = 1 + 0.1d$, which scales the base processing time once the start segment d is selected. In the formulation a position-based assignment structure is used: each machine l is endowed with a set of ordered positions $\mathcal{K} = \{1, \dots, C_2\}$, and each job is assigned to exactly one machine, one position and one configuration (subject to eligibility). Immediate-precedence variables x_{ijl} represent adjacency relationships on each machine, enabling the activation of setup times and configuration-change penalties. We summarize the notation used throughout the formulation in Table 1 and provide a formal definition of UPMS-SDS-NCS in the following.

Definition 1 (UPMS-SDS-NCS). *UPMS-SDS-NCS seeks to determine (i) the assignment of each job $i \in \mathcal{I}$ to one unrelated parallel machine $l \in \mathcal{L}$, one ordered position $k \in \mathcal{K}$, and one configuration $c \in \mathcal{C}$ (restricted to eligible triples $(i,l,c) \in \mathcal{E}$), (ii) the resulting sequence of jobs on each machine under sequence-dependent setup times, and (iii) the start and completion times (s_i, f_i) of all jobs. If job j immediately follows job i on machine l, the schedule must include a setup time T_{ijl}; if the configurations assigned to i and j differ, an additional configuration-change penalty* CFG *is incurred. Workforce dynamics are represented by three non-cyclic shift segments defined by boundaries H_1 and H_2; under the start-time based rule, the processing duration of job i is scaled by $\Gamma(d)$ where $d \in \mathcal{D}$ is the segment in which s_i lies. Jobs must respect release times R_i, and tardiness is measured relative to due dates D_i. The objective is to minimize total tardiness. To simplify notation, assignment variables are indexed only over feasible job–machine–configuration triples. That is, y_{ilkc} is defined only when $L_{ilc} = 1$ (equivalently, $(i,l,c) \in \mathcal{E}$), and any summation involving y is understood to be restricted to eligible indices. The multiplier $\Gamma(d)$ yields a non-preemptive approximation of shift-dependent productivity: once a job starts, its entire processing duration is determined by the segment d selected for its start time, even if processing overlaps later segments.*

Table 1. Nomenclature for the UPMS-SDS-NCS

Sets:		
$\mathcal{I} = \{1, \dots, \bar{I}\}$	jobs, indexed by i, j	
$\mathcal{L} = \{1, \dots, \bar{L}\}$	machines, indexed by l	
$\mathcal{C} = \{1, \dots, \bar{C}\}$	configurations, indexed by c	
$\mathcal{K} = \{1, \dots, C_2\}$	ordered positions on each machine, indexed by k	
$\mathcal{D} = \{0,1,2\}$	shift segments, indexed by d	
$\mathcal{E} = \{(i,l,c): L_{ilc} = 1\}$	eligible job–machine–config triples	
Parameters:		
P_{ilc}	processing time of job i on machine l under configuration c	
T_{ijl}	setup time between jobs i and j on machine l	
$L_{ilc} \in \{0,1\}$	eligibility: $= 1$ iff i can run on (l,c)	
R_i	release time of job i	
D_i	due date of job i	
CFG	configuration-change penalty between consecutive jobs	
C_1, C_2	min / max number of jobs processed per machine	
H_1, H_2	within-horizon shift boundaries defining three segments	
$\Gamma(d) = 1 + 0.1\,d$	start-time based rate multiplier for segment $d \in \{0,1,2\}$	
M	sufficiently large constant	

Decision variables:		
y_{ilkc}	$= 1$ if job i is assigned to machine l, position k, config c	
x_{ijl}	$= 1$ if job i immediately precedes job j on machine l	
w_{ijl}	$= 1$ if a configuration change occurs from i to j on machine l	
n_{id}	$= 1$ if job i starts in shift segment d	
s_i, f_i	start/completion time of job i	
u_i	tardiness of job i	

3.1 MILP formulation of the UPMS-SDS-NCS

We present a mixed-integer linear programming formulation consistent with the provided implementation. A position-based assignment structure is used: each machine l is endowed with C_2 ordered positions, and each job is assigned to exactly one position on exactly one machine (subject to eligibility). Immediate-precedence variables x_{ijl} represent adjacency on machine l. Temporal feasibility is enforced by disjunctive big-M constraints that activate setup and configuration-change penalties only when $x_{ijl} = 1$ and both jobs are assigned to the same machine. The objective minimizes total tardiness.

$$\min \quad \sum_{i \in \mathcal{I}} u_i \tag{1}$$

$$x_{iil} = 0 \qquad \forall i \in \mathcal{I},\ l \in \mathcal{L} \tag{2}$$

$$\sum_{l \in \mathcal{L}} \sum_{k \in \mathcal{K}} \sum_{c \in C} y_{ilkc} = 1 \qquad \forall i \in \mathcal{I} \tag{3}$$

$$\sum_{i \in \mathcal{I}} y_{ilkc} \le 1 \qquad \forall l \in \mathcal{L},\ k \in \mathcal{K},\ c \in C \tag{4}$$

$$C_1 \le \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}} \sum_{c \in C} y_{ilkc} \qquad \forall l \in \mathcal{L} \tag{5}$$

$$\sum \sum \sum \cdots \le C \qquad \forall l \in \mathcal{L} \tag{6}$$

$$x_{ijl} \ge \Big(\sum_{c \in C} y_{il,k-1,c} \Big) + \Big(\sum_{c \in C} y_{jl,k,c} \Big) - 1 \qquad \forall i \ne j,\ l \in \mathcal{L},\ k = 2, \ldots, C_2 \tag{10}$$

$$x_{ijl} + x_{jil} \le 1 \qquad \forall l \in \mathcal{L},\ i < j \tag{11}$$

$$w_{ijl} \ge x_{ijl} + \sum_{k \in \mathcal{K}} y_{ilkc_1} + \sum_{k \in \mathcal{K}} y_{jlkc_2} - 2 \qquad \forall i \ne j,\ l \in \mathcal{L},\ c_1 \ne c_2 \tag{12}$$

$$w_{ijl} \le x_{ijl} \qquad \forall i \ne j,\ l \in \mathcal{L} \tag{13}$$

$$s_i > R_i \qquad \forall i \in \mathcal{I} \tag{14}$$

$$s_i \le H_1 + M(1 - n_{i0}) \qquad \forall i \in \mathcal{I} \tag{16}$$

$$s_i \ge H_1 - M(1 - n_{i1}) \qquad \forall i \in \mathcal{I} \tag{17}$$

$$s_i \le H_2 + M(1 - n_{i1}) \qquad \forall i \in \mathcal{I} \tag{18}$$

$$s_i \ge H_2 - M(1 - n_{i2}) \qquad \forall i \in \mathcal{I} \tag{19}$$

$$s_j \ge f_i + T_{ijl} + \mathrm{CFG}\, w_{ijl} - M(1 - x_{ijl}) - M\Big(2 - \sum_{k,c} y_{ilkc} - \sum_{k,c} y_{jlkc}\Big) \qquad \forall i \ne j,\ l \in \mathcal{L} \tag{20}$$

$$f_i - s_i \le \sum_{l \in \mathcal{L}} \sum_{k \in \mathcal{K}} \sum_{c \in C} P_{ilc}\, \Gamma(d)\, y_{ilkc} + M(1 - n_{id}) \qquad \forall i \in \mathcal{I},\ d \in \mathcal{D} \tag{21}$$

$$f_i - s_i \ge \sum_{l \in \mathcal{L}} \sum_{k \in \mathcal{K}} \sum_{c \in C} P_{ilc}\, \Gamma(d)\, y_{ilkc} - M(1 - n_{id}) \qquad \forall i \in \mathcal{I},\ d \in \mathcal{D} \tag{22}$$

$$u_i \ge f_i - D_i \qquad \forall i \in \mathcal{I} \tag{23}$$

$$y_{ilkc}, x_{ijl}, w_{ijl}, n_{id} \in \{0, 1\} \qquad \forall i, j \in \mathcal{I},\ l \in \mathcal{L},\ k \in \mathcal{K},\ c \in C,\ d \in \mathcal{D} \tag{24}$$

$$s_i, f_i, u_i \ge 0 \qquad \forall i \in \mathcal{I} \tag{25}$$

Objective function (1) minimizes total tardiness. Constraints (2) forbid self-precedence. Assignment constraints (3) ensure each job is processed exactly once on an eligible machine–configuration–position. Position capacity (4), monotone position usage (7), and per-machine load bounds (5)-(6) match the position-based structure used in the code. Precedence variables are linked to assignments via (8)-(9) and are *activated* by adjacency through (10): if job i occupies position $k-1$ and job j occupies position k on the same machine l, then $x_{ijl}=1$ must hold, ensuring setup and configuration penalties are triggered. Equation (11) enforces anti-symmetry, prohibiting simultaneous reciprocal relationships between indices i and j on the same machine. Configuration changes are detected with (12)-(13), so CFG

is added when consecutive jobs use different configurations. Release dates are enforced by (14). Shift-segment selection (15)-(19) assigns each job to exactly one of the three segments defined by (s_1, s_2) and restricts s_i accordingly (segment 0: $s_i \le s_1$; segment 1: $s_1 \le s_i \le s_2$; segment 2: $s_i \ge s_2$). Temporal feasibility with setup and configuration penalties is enforced by (20), which is active only when $x_{ijl}=1$ and both jobs are assigned to machine l. Processing durations are defined by (21)-(22) using the start-time multiplier $\Gamma(d)=1+0.1d$ selected by n_{id}. Finally, tardiness is defined by (23) as $u_i=\max\{0, f_i-D_i\}$ due to the nonnegativity of u_i .

4. Solution Method

This section presents the solution technique developed for the UPMS-SDS-NCS. Given the strong combinatorial complexity induced by unrelated parallel machines, sequence-dependent setups, configuration reconfiguration penalties, and shift-dependent processing rates, exact optimization methods become computationally intractable for realistically sized instances. We therefore develop an ALNS as the primary solution approach. The proposed ALNS constitutes the core methodological contribution of this study. The ALNS follows a destroy–repair paradigm with adaptive operator selection, feasibility-aware simulation-based evaluation, and simulated-annealing-style acceptance. The algorithm operates directly on machine-wise job sequences and explicitly simulates temporal feasibility under release times, setup times, configuration changes, and start-time-based shift multipliers.

4.1 Solution Representation

A solution S is represented as a collection of machine schedules

$$S=\{S_1, S_2, \dots, S_{|\mathcal{L}|}\},$$

where each machine $l \in \mathcal{L}$ is associated with an ordered sequence

$$S_l=\left(v_{l,1}, v_{l,2}, \dots, v_{l,n_l}\right).$$

Each element $v_{l,k}$ represents a scheduled job $i \in \mathcal{I}$. Each job in the sequence is further associated with a configuration choice $c_{l,k} \in \mathcal{C}$ such that $(i,l,c_{l,k}) \in \mathcal{E}$. Each job $i \in \mathcal{I}$ appears *exactly once* across all machine sequences $\{S_l\}$, ensuring a one-to-one assignment between jobs and machine positions. The representation enforces eligibility by requiring that every scheduled triple $(i,l,c_{l,k})$ belongs to the feasible set $\mathcal{E}$. The sequence order within each S_l defines both the processing order and the immediate-precedence relationships used to evaluate sequence-dependent setup times T_{ijl} and configuration-change penalties Δ. Start times and completion times are not stored explicitly in the solution representation but are computed through forward simulation during evaluation.

4.2 Solution Evaluation

Given a candidate solution S, feasibility and objective value are evaluated by simulating each machine schedule in chronological order. For each machine l, processing starts from time zero and proceeds sequentially through the jobs in S_l. For a given job $v_{l,k}=i$, the start time s_i is computed as

$$s_i=\max\{R_i, f_{l,k-1}+T_{jil}+\Delta\cdot\mathbb{I}[c_{l,k-1}\neq c_{l,k}]\},$$

where $f_{l,k-1}$ is the completion time of the previous job on machine l, and $\mathbb{I}[\cdot]$ is an indicator for configuration change (with $j=v_{l,k-1}$ denoting the predecessor job in the sequence). The processing duration is determined by the machine–configuration pair $(l,c_{l,k})$ and the shift segment in which the job starts. Let d denote the shift interval such that $s_i \in [\underline{W}_d, \overline{W}_d]$. The effective processing time is

$$p_i^{\text{eff}}=P_{ilc_{l,k}}\cdot\Gamma(H_d).$$

The completion time is then $f_i=s_i+p_i^{\text{eff}}$, and the job completion time is defined as $F_i=f_i$. Tardiness is computed relative to the due date D_i as

$$u_i=\max\{0, F_i-D_i\}.$$

The objective value of a solution is

$$f(S) = \sum_{i \in \mathcal{J}} u_i.$$

Infeasibilities arising from eligibility violations, duplicate assignments, or inconsistent sequencing are prevented by construction in the repair phase and checked during evaluation. Any remaining infeasibility is penalized through large additive penalties to the objective value.

4.3 Initial Solution Construction

The ALNS is initialized with a greedy constructive heuristic. Jobs are first sorted in nondecreasing order of release times (ties broken by earlier due dates). Each job is tentatively assigned to the machine–configuration pair $(l, c) \in \mathcal{E}$ that yields the earliest feasible completion time under the current partial schedule. The job is then inserted at the best feasible position within the selected machine sequence, taking into account sequence-dependent setup times, configuration-change penalties, and shift-dependent processing rates. This procedure yields a feasible initial solution S_0 that respects release dates and eligibility constraints, providing a structured starting point for the ALNS.

4.4 Destroy Operators

The ALNS employs multiple destroy operators that remove a subset of jobs from the current solution to create temporal and structural slack. The destruction rate ρ controls the fraction of jobs affected. The main destroy operators include:

- **Random Job Removal:** Randomly removes a fraction ρ of jobs, ignoring their machine assignment and sequence position.
- **Tardiness-Based Removal:** Targets jobs with the largest tardiness contributions, removing those that most degrade the objective value.
- **Expensive Shift Removal:** Selects one or more jobs that start in the most expensive shift segments (according to $\Gamma(H_d)$) and removes them to encourage rescheduling into less costly intervals.

These operators are designed to disrupt different structural aspects of the schedule, including sequencing and machine assignment.

4.5 Repair Operators

After destruction, the repair phase reinserts removed jobs using feasibility-aware heuristics. Let $\mathcal{U}$ denote the set of unassigned jobs. The repair operators evaluate feasible insertion positions across machines and eligible configurations and reinstate jobs while respecting all feasibility rules. The main repair operators include:

- **Greedy Insertion:** For each job $i \in \mathcal{U}$, the operator evaluates feasible insertion positions across all machines and configurations. The job is inserted at the position that minimizes the resulting total tardiness.
- **Regret-k Insertion:** For each job, the operator computes the k best insertion options (in terms of objective deterioration). The regret value is defined as the difference between the best and the k-th best option. Jobs with the largest regret are inserted first, prioritizing those whose insertion opportunities are most fragile.
- **Earliest-Completion Insertion:** For each job $i \in \mathcal{U}$, the operator evaluates feasible insertion positions across all machines and configurations. The job is inserted at the position that minimizes its completion time F_i.
- **Random Insertion:** Each job $i \in \mathcal{U}$ is inserted at a randomly chosen feasible position on a compatible machine to ensure diversification.

4.6 Local Search (q)

To intensify the search with negligible computational overhead, we apply a local search that *only* optimizes the configuration decisions while keeping the current job-to-machine assignments and sequences fixed. For each machine l, given the ordered job list $S_l = (v_{l,1}, \dots, v_{l,n_l})$, the goal is to choose configurations $c_{l,k} \in \mathcal{C}$ (subject to eligibility $(v_{l,k}, l, c_{l,k}) \in \mathcal{E}$) that yield the best schedule in terms of time propagation along the sequence. This subproblem is solved by a Viterbi-style Dynamic Program (DP). The DP state is the configuration of the current job, and the DP value stores the earliest completion time achieved up to position k. For the transition from the previous job $j = v_{l,k-1}$ with configuration c' to the current job $i = v_{l,k}$ with configuration c, we simulate

$$s_i = \max\{R_i,\ f_j + T_{jil} + \Delta \cdot \mathbb{I}[c' \neq c]\}, \qquad f_i = s_i + P_{ilc} \cdot \Gamma(H_{d(s_i)}),$$

where T_{jil} is the sequence-dependent setup time, Δ is the configuration-change penalty, and $\Gamma(H_{d(s_i)})$ is the shift-dependent multiplier determined by the start time. The DP selects, for each position k and configuration c, the predecessor configuration c' that minimizes f_i. After the forward pass, we backtrack once to assign the best

configuration chain to all jobs in S_l. Since this procedure runs in $\mathcal{O}(\sum_{l\in\mathcal{L}} n_l\, \bar{C}^2)$ in the worst case, it is computationally cheap and can be applied frequently (e.g., after each repair) to improve solutions by exploiting better configuration sequences under setup, reconfiguration, release times, and shift effects.

4.7 Overall ALNS Framework

Algorithm 1 summarizes the overall ALNS procedure for the UPMS-SDS-NCS.

Algorithm 1 Adaptive Large Neighborhood Search for the UPM–RC

```
Construct an initial feasible solution S_0
S ← S_0, S* ← S_0
Initialize destroy and repair operator weights
while Stopping condition met do
    Select destroy operator d and repair operator r adaptively
    Choose destruction rate ρ
    S̃ ← d(S, ρ)
    S' ← r(S̃)
    S'' ← q(S')
    Evaluate S'' via simulation and compute f(S'')
    if f(S'') < f(S*) then
        S* ← S''
    end if
    if acceptance criterion holds then
        S ← S''
        Update operator weights
    end if
end while
return S*
```

The resulting ALNS framework is specifically tailored to the UPMS-SDS-NCS. By combining adaptive large-scale neighborhood exploration with detailed simulation-based evaluation, the algorithm effectively balances intensification and diversification while explicitly accounting for machine heterogeneity, reconfiguration effects, and shift-dependent processing rates.

5. Computational Experiments

We evaluate the effectiveness of the proposed ALNS algorithm by comparing its performance against an adapted version of the SHA heuristic introduced by (Ekici et al. 2019). The experiments are conducted on synthetically generated instances that are designed to closely resemble the modified real-life instances presented in (Ekici et al. 2019). Unlike the original monthly planning horizon used in that study, our instances are constructed for a daily scheduling environment, where all time-related parameters are converted into minutes according to the same modeling rules. A total of 36 instances are generated and classified into three categories—small, medium, and large—based on the number of jobs and machines. All instances incorporate key characteristics of the problem setting, including machine–job eligibility constraints, sequence-dependent setup times, shift-based processing, and reconfiguration requirements.

In the small and medium instance classes, the number of jobs varies between 10 and 50, while the number of machines ranges from 2 to 8. The number of configurations is fixed and set to two for all instances in these classes. For the large instance class, the number of jobs ranges between 100 and 200, and the number of machines varies between 8 and 20. Job release dates and due dates are generated consistently with the problem setting and scaled to the daily planning horizon. ll computational experiments were conducted on a 64-bit Windows 10 Pro system running on a virtual machine equipped with an AMD EPYC 7662 processor (8 cores, 1.99 GHz) and 29.3 GB of RAM. The computational experiments are performed under identical hardware and software conditions to ensure a fair comparison between the proposed ALNS approach and the adapted SHA heuristic. In Table 2 and Table 3, we compare the performance of the proposed ALNS algorithm and the adapted SHA heuristic against an exact MIP model implemented in Gurobi for the small and medium instance classes. For these instances, the Gurobi model is executed with a time limit of 3600 seconds. The ALNS algorithm is run with a fixed time limit of 1800 seconds for all instances. The SHA heuristic is modified to be consistent with our problem setting by explicitly incorporating configuration assignment decisions for each machine. Given a complete assignment of jobs and configurations to machines, single-machine scheduling problems are solved independently and in parallel, subject to a maximum total computation time of 1800 seconds across all

machines. The ALNS algorithm employs an adaptive weight adjustment mechanism based on the following reward parameters: $\sigma_1 = 10$ for generating a solution that improves upon the global best solution, $\sigma_2 = 5$ for improving the current local solution, and $\sigma_3 = 2$ for generating an accepted but non-improving solution by Simulated Annealing.$\sigma_4 = 0.1$ for non-improving rejected solution by Simulated Annealing. These parameters are selected to balance intensification and diversification during the search process. To quantify the relative performance of the algorithms, we measure the improvement with respect to a benchmark solution using the following metric:

$$\text{Improvement} = \frac{\text{Benchmark}_{\text{obj}} - \text{Algorithm}_{\text{obj}}}{\text{Benchmark}_{\text{obj}}} \times 100\%.$$

For the small and medium instance classes, the benchmark solution is obtained from the Gurobi model shown in Table 2. The results indicate that the adapted SHA heuristic produces solutions that are, on average, 63.7% worse than the Gurobi benchmark in small instances. In contrast, the proposed ALNS algorithm achieves an average improvement of approximately 7.3% over the benchmark across small and medium instances.

Table 2. Performance comparison of the exact model, ALNS, and SHA under small and medium instance sizes

#Jobs	#Machines	Gap (%)	Model Time	ALNS Time	SHA Time	ALNS vs Model (%)	SHA vs Model (%)
10	2	0.00	2650.10	1800.00	1.71	0%	0%
10	2	0.00	50.57	1800.00	1.70	0%	-43%
10	2	0.00	29.05	1800.00	4.93	0%	-46%
10	3	0.00	35.06	1800.00	1.98	0%	-313%
10	3	0.00	17.83	1800.00	2.01	0%	-235%
10	3	0.00	36.52	1800.00	1.63	0%	-247%
30	3	99.74	3607.49	1800.01	1803.01	2%	-6%
30	3	99.18	3610.65	1800.03	1802.78	4%	-8%
30	3	98.93	3608.66	1800.01	1802.79	5%	-8%
30	5	99.89	3610.82	1800.03	248.39	12%	-52%
30	5	99.77	3611.68	1800.06	24.13	5%	-29%
30	5	95.93	3610.87	1800.00	1802.05	8%	-115%
50	5	99.36	3632.93	1800.52	1803.16	10%	7%
50	5	99.34	3628.00	1800.24	1803.20	11%	4%
50	5	99.86	3624.93	1800.43	1804.47	15%	6%
50	8	98.57	3635.11	1800.23	283.43	20%	-40%
50	8	98.82	3633.37	1800.15	1802.38	21%	-20%
50	8	99.64	3635.01	1800.31	35.26	19%	-2%
Average	–	**66.1**	**2570.5**	**1800.1**	**834.9**	**7.3%**	**-63.7%**

For the large instance class, the Gurobi model fails to identify feasible solutions within the imposed time limit due to the increased problem complexity. Consequently, the adapted SHA heuristic is used as the benchmark for large instances. Under this setting, the ALNS algorithm consistently outperforms SHA, achieving an average improvement of approximately 6.17% shown in Table 3. We observe that ALNS outperforms SHA in not only solution quality but also in CPU time performance.

Table 3. Performance comparison of the ALNS and SHA in large instance sizes

#Jobs	#Machines	ALNS Time	SHA Time	ALNS vs SHA(%)
100	8	1802.088363	3605.271	0%
100	8	1800.405791	2072.223	2%
100	8	1800.399607	3604.211	-2%
100	10	1800.046129	1803.565	7%
100	10	1801.693394	1817.13	11%
100	10	1800.475527	1804.396	14%
150	13	1808.920488	3604.795	2%
150	13	1807.742559	3605.848	0%
150	13	1800.831765	3605.254	7%
150	15	1807.769974	3604.825	9%
150	15	1803.542738	3604.322	8%
150	15	1802.308441	3607.148	14%
200	17	1806.769882	5404.9	3%
200	17	1805.228031	3792.298	2%
200	17	1807.142052	3856.275	1%
200	20	1800.266101	3632.457	13%
200	20	1814.416893	3667.388	12%
200	20	1806.089840	3641.950	8%
Average	–	**1804.2**	**3351.9**	**6.17%**

6. Conclusion

This study addressed the Unrelated Parallel Machine Scheduling Problem (UPMSP) with sequence-dependent setup times and shift-dependent workforce constraints to minimize total tardiness. To solve this complex NP-hard problem, we developed a Mixed-Integer Linear Programming (MILP) model for small instances and an Adaptive Large Neighborhood Search (ALNS) algorithm for larger datasets. Computational experiments on 36 generated instances demonstrated the superior performance of the proposed ALNS. In small and medium instances, the ALNS achieved an average improvement of 7% over the benchmark solutions provided by the exact solver, significantly outperforming an adapted Sequential Heuristic Approach (SHA). For large-scale instances where the exact solver failed to find feasible solutions, the ALNS consistently surpassed the SHA heuristic by an average of 6% in solution quality while maintaining high computational efficiency. These results confirm that the proposed ALNS is a robust and scalable method for solving shift-constrained scheduling problems. Future research will explore stochastic processing times and multi-objective formulations balancing tardiness with operational costs.

References

Afzalirad, Mojtaba, and Javad Rezaeian. "Resource-Constrained Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times, Precedence Constraints and Machine Eligibility Restrictions." *Computers & Industrial Engineering* 98: 40–52. 2016.

Allahverdi, Ali.. "The Third Comprehensive Survey on Scheduling Problems with Setup Times/Costs." *European Journal of Operational Research* 246 (2): 345–78. 2015

Berthier, Alice, Alice Yalaoui, Hicham Chehade, Farouk Yalaoui, Lionel Amodeo, and Christian Bouillot. "Unrelated Parallel Machines Scheduling with Dependent Setup Times in Textile Industry." *Computers & Industrial Engineering* 174: 108736. 2022.

Chen, Jeng-Fung. "Scheduling on Unrelated Parallel Machines with Sequence-and Machine-Dependent Setup Times and Due-Date Constraints." *The International Journal of Advanced Manufacturing Technology* 44 (11): 1204–12. 2009.

Dou, Jianping, Jun Li, Dan Xia, and Xia Zhao. "A Multi-Objective Particle Swarm Optimisation for Integrated Configuration Design and Scheduling in Reconfigurable Manufacturing System." *International Journal of Production Research* 59 (13): 3975–95. 2021.

Đurasević, Marko, and Domagoj Jakobović. "Heuristic and Metaheuristic Methods for the Parallel Unrelated Machines Scheduling Problem: A Survey." *Artificial Intelligence Review* 56 (4): 3181–289. 2023.

Ekici, Ali, Milad Elyasi, Okan Örsan Özener, and Merve Burcu Sarıkaya. "An Application of Unrelated Parallel Machine Scheduling with Sequence-Dependent Setups at Vestel Electronics." *Computers & Operations Research* 111: 130–40. 2019.

Elyasi, Milad, Yagmur Selenay Selcuk, O Örsan Özener, and Elvin Coban. "Imperialist Competitive Algorithm for Unrelated Parallel Machine Scheduling with Sequence-and-Machine-Dependent Setups and Compatibility and Workload Constraints." *Computers & Industrial Engineering* 190: 110086. 2024.

Fang, Wei, Haolin Zhu, and Yi Mei. "Hybrid Meta-Heuristics for the Unrelated Parallel Machine Scheduling Problem with Setup Times." *Knowledge-Based Systems* 241: 108193. 2022.

Lee, Young Hoon, and Michael Pinedo. "Scheduling Jobs on Parallel Machines with Sequence-Dependent Setup Times." *European Journal of Operational Research* 100 (3): 464–74. 1997.

Li, Debiao, Jing Wang, Rui Qiang, and Raymond Chiong. "A Hybrid Differential Evolution Algorithm for Parallel Machine Scheduling of Lace Dyeing Considering Colour Families, Sequence-Dependent Setup and Machine Eligibility." *International Journal of Production Research* 59 (9): 2722–38. 2021.

Logendran, Rasaratnam, Brent McDonell, and Byran Smucker.. "Scheduling Unrelated Parallel Machines with Sequence-Dependent Setups." *Computers & Operations Research* 34 (11): 3420–38. 2007

Mecler, Davi, Victor Abu-Marrul, Rafael Martinelli, and Arild Hoff. "Iterated Greedy Algorithms for a Complex Parallel Machine Scheduling Problem." *European Journal of Operational Research* 300 (2): 545–60. 2022.

Pinedo, Michael L. *Scheduling: Theory, Algorithms, and Systems*. 5th ed. Springer. 2016.

Ropke, Stefan, and David Pisinger. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows." *Transportation Science* 40 (4): 455–72. 2006.

Santoro, Miguel Cezar, and Leonardo Junqueira. "Unrelated Parallel Machine Scheduling Models with Machine Availability and Eligibility Constraints." *Computers & Industrial Engineering* 179: 109219. 2023.

Uribe, Alejandro, Urtzi Otamendi, Arkaitz Artetxe, and Juan Carlos Rivera. "Mathematical Formulations for the Scheduling of Unrelated Parallel Machines Considering Machine Availability and Eligibility." *Expert Systems with Applications*, 127773. 2025.

Vallada, Eva, and Rubén Ruiz. "A Genetic Algorithm for the Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times." *European Journal of Operational Research* 211 (3): 612–22. 2011.

Xie, Fulong, Kai Li, Jianfu Chen, Wei Xiao, and Tao Zhou. "An Adaptive Large Neighborhood Search for Unrelated Parallel Machine Scheduling with Setup Times and Delivery Times." *Computers & Operations Research* 177: 106976. 2025.

Xu, Jianyou, Chin-Chia Wu, Yunqiang Yin, Chuanli Zhao, Yi-Tang Chiou, and Win-Chin Lin. "An Order Scheduling Problem with Position-Based Learning Effect." *Computers & Operations Research* 74: 175–86. 2016.

Yanıkoğlu, İhsan, and Tonguc Yavuz. "Branch-and-Price Approach for Robust Parallel Machine Scheduling with Sequence-Dependent Setup Times." *European Journal of Operational Research* 301 (3): 875–95. 2022.

Biographies

Demiray Demir is a Master's student in Data Science at Özyeğin University, Istanbul, Turkey. He completed his Bachelor's degree in Industrial Engineering at Özyeğin University. His research interests lie at the intersection of Machine Learning, Reinforcement Learning, and Operations Research, with a particular focus on combinatorial optimization problems. He works on the development and analysis of both exact and heuristic solution approaches for complex decision-making and scheduling problems.

Milad Elyasi is an Assistant Professor in the Industrial Engineering Department at Özyeğin University, Istanbul, Turkey. He received his Ph.D. in Industrial Engineering and his research interests include operations research, mathematical optimization, transportation and energy systems, and decision analytics. His work focuses on the development of optimization models and algorithms for complex industrial and energy-related problems, including sustainable manufacturing, energy-efficient system design, and large-scale optimization under uncertainty. He has published in international journals and conference proceedings and actively supervises graduate research in optimization and applied operations research.