

Integrating Workforce Allocation and Parallel Machine Scheduling with Setup- and Release-Dependent Constraints

Betül Kılınç

Faculty of Engineering and Natural Sciences
Industrial Engineering Department
Ankara Yıldırım Beyazıt University, Ankara, Türkiye
betulkilinc18@gmail.com.tr

Cihat Öztürk

Asst. Prof, Faculty of Engineering and Natural Sciences
Industrial Engineering Department
Ankara Yıldırım Beyazıt University, Ankara, Türkiye
cozturk@aybu.edu.tr

Abstract

This study investigates a complex scheduling problem involving parallel machines with sequence-dependent setup times, job release dates, and worker-based human resource constraints. Unlike classical scheduling models that consider only machines as limited resources, this problem reflects real industrial settings where both machines and workers must be coordinated simultaneously. Each job can only start after its release date and requires a fixed number of workers from a limited workforce. To ensure workers are not assigned to overlapping jobs, explicit disjunctive constraints are incorporated into the model. Sequence-dependent setup times are also considered, meaning that setup durations vary according to job order. This significantly increases problem complexity while accurately representing industries such as manufacturing and healthcare. The integration of release dates, workforce requirements, and setup-dependent sequencing makes the problem NP-hard. To solve the problem, a mixed-integer linear programming (MILP) model is proposed. The model assigns jobs to machines, defines sequencing relations, and integrates processing, setup, and workforce constraints. The objective minimizes a weighted combination of makespan and total completion time, balancing operational efficiency and flow performance. The commercial solver Gurobi is used to obtain optimal solutions for small and medium-sized instances. For larger problems, a General Variable Neighborhood Search (GVNS) heuristic is developed. The proposed GVNS effectively explores multiple neighborhoods and incorporates release date and workforce constraints. Computational results show that GVNS produces near-optimal solutions with significantly lower computational times, making it suitable for large-scale industrial scheduling applications.

Keywords

Parallel machine scheduling, Sequence-dependent setup times, Workforce allocation, General Variable Neighborhood Search

1. Introduction

Today, with the effects of globalization and rapid technological advancements, competition in manufacturing and service industries has intensified, making speed, flexibility, and cost efficiency critical objectives for firms. Achieving these objectives requires the simultaneous planning of multiple organizational constraints such as machines, workforce, and raw materials. Traditional scheduling approaches generally consider only a limited number of factors

while assuming other constraints to be unlimited. However, the necessity of responding rapidly to dynamically changing customer demands restricts the practical applicability of these idealized approaches in real-life environments. Particularly in job-shop production systems, continuously changing operational steps and workforce limitations make planning processes significantly more challenging. Parallel Machine Scheduling (PMS) problems constitute an important research area due to their widespread applications in sectors such as electronics assembly, healthcare services, and logistics. Classical machine-centered models are insufficient in addressing sequence-dependent setup times, release dates, and workforce-related constraints. The variation of transition times according to job sequences and the inability of workers to perform multiple tasks simultaneously negatively affect the realism of solutions when such constraints are excluded from the model.

This study presents a Mixed-Integer Linear Programming (MILP) model that simultaneously optimizes machine assignment and workforce allocation under sequence-dependent setup times and release date constraints. The proposed model assigns jobs to machines while enforcing precedence relationships, associate's completion times with setup and processing durations, and prevents personnel conflicts. Total completion time and weighted combinations were employed as objective functions, and the model generated optimal solutions for small- and medium-scale problem instances using Gurobi.

Due to the increasing computational burden in large-scale problems, a Generalized Variable Neighborhood Search (GVNS) heuristic approach was also developed. This algorithm explores the solution space through a two-phase search mechanism, enabling the efficient solution of large-scheduling problems.

With the diversification of customer demands and the increasing customization of products, highly variable structures have emerged in job-shop production systems. This situation introduces different testing procedures, setup requirements, and auxiliary equipment needs, causing classical planning approaches to become insufficient.

In the examined production environment, there are parallel and identical machines, and theoretically each product can be processed on any machine. However, in practice, a product-specific setup process involving apparatus replacement, calibration, and software installation is mandatory for each product. In the current system, jobs are assigned according to arrival order or simple priority rules, while setup times and labor utilization are not systematically optimized. This leads to increased total setup times, longer machine idle periods, and workforce bottlenecks. Set-up activities constitute the most critical component of the system. An increase in the number of transitions between products requires additional setup time, whereas scheduling related products consecutively can reduce these durations. Therefore, the production sequence becomes the primary factor determining system performance. On the other hand, setup operations are conducted by a limited and indivisible workforce, meaning that a worker cannot operate on more than one machine simultaneously. Even if a machine is idle, a new job cannot be started when the required personnel are unavailable, effectively transforming workforce availability into a real capacity constraint. In addition, the unavailability of product-specific setup equipment and the inability to release certain jobs before a specified starting time due to material supply or technical approval processes make the problem dynamic and multidimensional.

The considered problem requires the simultaneous optimization of machine assignment, job sequencing, and workforce allocation under sequence-dependent setup times, limited workforce capacity, equipment availability, and time-related constraints. Since each decision component directly affects the others, an integrated and optimization-based planning approach is required.

2. Literature Review

The literature on Parallel Machine Scheduling (PMS) has shown continuous development over the last three decades, evolving from fundamental mathematical models toward hybrid, data-driven, and highly constrained structures that reflect the full operational complexity of real-world manufacturing systems.

Schutten and Leussink (1996) addressed identical parallel machines with release dates and family-dependent setup times by employing a Branch-and-Bound algorithm integrated with Carlier's lower bounds and enhanced through pruning strategies to minimize maximum tardiness. Rojanasoonthon et al. (2003) maximized the number of processed jobs by combining dynamic programming with a GRASP-based algorithm under time-window and antenna-availability constraints derived from NASA satellite control systems. Castillo et al. (2009) investigated multi-objective

interactions between cost minimization and service-level maximization in workforce scheduling through DEA and interactive search procedures.

The optimization of sequence-dependent setup times (SDST) increased the complexity of scheduling problems to the NP-hard class, thereby accelerating the development of heuristic methodologies. Behnamian et al. (2009) proposed a three-phase hybrid algorithm combining Ant Colony Optimization (ACO), Simulated Annealing (SA), and Variable Neighborhood Search (VNS). Avalos-Rosales et al. (2014) developed new MILP formulations for unrelated parallel machine problems with SDST, achieving exact solutions up to four times faster than previous approaches.

By the mid-2010s, hybrid methodologies and constraint diversity had reached their peak. Lin and Hsieh (2014) introduced an EMA-based Iterated Hybrid Metaheuristic, while Wang et al. (2016) proposed a hybrid approach integrating Estimation of Distribution Algorithms (EDA) with Iterated Greedy (IG). Thevenin et al. (2017) modeled job splitting and incompatibility as a graph-coloring problem, whereas Zhang and Lu (2016) introduced the concept of “rejection penalty” into the literature. Costa et al. (2016) optimized team transitions using a hybrid Genetic Algorithm. Recent studies have increasingly focused on industrial applications, uncertainty, and large-scale problems. Akyol Ozer and Sarac (2017) integrated MILP into a matheuristic framework, while Ekici et al. (2019) modeled 15 assembly lines of Vestel using Tabu Search. Cheng et al. (2020) developed the RF-HABC algorithm, which estimates setup times using the Random Forest method. Yunusoglu and Topaloglu Yildiz (2022) demonstrated the superiority of Constraint Programming over MILP approaches. Finally, Li et al. (2022) proposed efficient MILP formulation and a three-phase matheuristic capable of solving large-scale problems involving up to 3,000 jobs. Đurasević and Jakobović (2022) summarized the methodological evolution through a systematic survey covering 245 studies and provided a comprehensive roadmap for future research.

3. Problem Formulation

The developed mathematical model simultaneously optimizes job assignment and sequencing on parallel test machines under product-specific setup times, indivisible operator resources, and equipment availability constraints. Beyond the scope of classical parallel machine scheduling, the proposed integrated structure incorporates workforce-based and set-up-dependent considerations to minimize the total completion time.

Sets and Indices

- $J_n = \{j_1, \dots, j_n\}$: Real jobs.
- $J = \{j_0\} \cup J_n$: Dummy starting job with added job j_0 .
- $K = \{k_1, \dots, k_m\}$: Parallel machines.
- $H = \{r_1, \dots, r_R\}$: Persons (total capacity R).

Parameters

- $p_{jk} \geq 0$: The processing time of job $j \in J_n$ in $k \in K'$
- $s_{ijk} \geq 0$: Setup time $i \rightarrow j$ on machine k ($i \in J, j \in J_n, i \neq j$),
- $\delta_{jk} \in \{0,1\}$: Eligibility (appointability),
- $r_j \geq 0$: Release time ($j \in J_n$),
- $h_j \in \{1,2,3\}$: Person need ($j \in J_n$),
- $R \in \mathbb{Z}_{\geq 1}$: Total number of people,
- $M_k > 0$: Big $-M$ (machine specific)
- $T > 0$: Time horizon (in code $T = \max_k M_k$),
- $w_1, w_2 \geq 0$: Purpose weight (in code $w_1 = 1, w_2 = 0$).

Decision Variables

- $Y_{jk} \in \{0,1\}$: $j \in J_n$'s assignment of the job to $k \in K$
- $X_{ijk} \in \{0,1\}$: In k , the arc $i \rightarrow j$ is ($i \in J, j \in J_n, i \neq j$),
- $C_{jk} \geq 0$: The completion time of the job $j \in J$ on $k \in K$,
- $S_j \geq 0$: Process start for $j \in J_n$ (after setup),
- $F_j \geq 0$: $j \in J_n$ is the completion time of the task,
- $U_{jk} \in [0, n]$: MTZ sequence variable ($j \in J_n, k \in K$),
- $Z_k \in \{0,1\}$: Is the machine k being used,

$C^{max} \geq 0$: Makespan,
 $W_{jr} \in \{0,1\}$: If person $r \in H$ is assigned to $j \in J_n$, then 1,
 $B_{ij}^{(r)} \in \{0,1\}$: (i,j) the same person in r if i first then j then 1.

Objective Function

$$\min Z = w_1 C^{max} + w_2 \sum_{j \in J_n} \sum_{k \in K} C_{jk} \quad (1)$$

Restrictions:

$$\sum_{k \in K} Y_{jk} = 1 \quad \forall j \in J_n \quad (2)$$

$$\sum_{\substack{i \in J \\ i \neq j}} X_{ijk} = Y_{jk} \quad \forall j \in J_n, \forall k \in K. \quad (3)$$

$$\sum_{\substack{j \in J_n \\ j \neq i}} X_{ijk} \leq Y_{ik} \quad \forall i \in J_n, \forall k \in K \quad (4)$$

$$\sum_{j \in J_n} X_{j_0jk} = Z_k \quad \forall k \in K, \quad (5)$$

$$\sum_{j \in J_n} Y_{jk} \geq Z_k \quad \forall k \in K, \quad (6)$$

$$\sum_{j \in J_n} Y_{jk} \leq nZ_k \quad \forall k \in K, \quad (7)$$

$$\sum_{\substack{j \in J_n, j \in J_n \\ i \neq j}} X_{ijk} = \sum_{j \in J_n} Y_{jk} - Z_k \quad \forall k \in K. \quad (8)$$

$$C_{j_0k} = 0 \quad \forall k \in K, \quad (9)$$

$$C_{jk} \geq s_{j_0jk} + p_{jk} - M_k(1 - X_{j_0jk}), \quad \forall j \in J_n, \forall k \in K, \quad (10)$$

$$C_{jk} \leq s_{j_0jk} + p_{jk} + M_k(1 - X_{j_0jk}), \quad \forall j \in J_n, \forall k \in K, \quad (11)$$

$$S_j \geq s_{j_0jk} - M_k(1 - X_{j_0jk}), \quad \forall j \in J_n, \forall k \in K, \quad (12)$$

$$S_j \leq s_{j_0jk} + M_k(1 - X_{j_0jk}), \quad \forall j \in J_n, \forall k \in K, \quad (13)$$

$$C_{jk} \geq C_{ik} + s_{ijk} + p_{jk} - M_k(1 - X_{ijk}), \quad \forall i \in J_n, i \neq j, \forall j \in J_n, \forall k \in K, \quad (14)$$

$$C_{jk} \leq C_{ik} + s_{ijk} + p_{jk} + M_k(1 - X_{ijk}), \quad \forall i \in J_n, i \neq j, \forall j \in J_n, \forall k \in K, \quad (15)$$

$$S_j \geq C_{ik} + s_{ijk} - M_k(1 - X_{ijk}), \quad \forall i \in J_n, i \neq j, \forall j \in J_n, \forall k \in K, \quad (16)$$

$$S_j \leq C_{ik} + s_{ijk} + M_k(1 - X_{ijk}), \quad \forall i \in J_n, i \neq j, \forall j \in J_n, \forall k \in K. \quad (17)$$

$$F_j \geq C_{jk} - M_k(1 - Y_{jk}) \quad \forall j \in J_n, \forall k \in K, \quad (18)$$

$$F_j \leq C_{jk} + M_k(1 - Y_{jk}) \quad \forall j \in J_n, \forall k \in K, \quad (19)$$

$$C_{jk} \leq M_k Y_{jk} \quad \forall j \in J_n, \forall k \in K \quad (20)$$

$$C^{max} \geq C_{jk} \quad \forall j \in J, \forall k \in K. \quad (21)$$

$$Y_{jk} \leq \delta_{jk} \quad \forall j \in J_n, \forall k \in K. \quad (22)$$

$$U_{jk} \leq nY_{jk}, \quad U_{jk} \geq Y_{jk} \quad \forall j \in J_n, \forall k \in K, \quad (23-24)$$

$$U_{jk} \geq U_{ik} + 1 - n(1 - X_{ijk}) \quad \forall i \in J_n, i \neq j, \forall j \in J_n, \forall k \in K \quad (25)$$

$$S_j \geq r_j \quad \forall j \in J_n, \quad (26)$$

$$\sum_{r \in H} W_{jr} = h_j \quad \forall j \in J_n, \quad (27)$$

(i<j), $r \in H$ için;

$$S_j \geq F_i - T \left(1 - B_{ij}^{(r)} + (1 - W_{ir}) + (1 - W_{jr}) \right) \quad \forall i, j \in J_n, i < j, \forall r \in H, \quad (28)$$

$$S_i \geq F_j - T \left(B_{ij}^{(r)} + (1 - W_{ir}) + (1 - W_{jr}) \right) \quad \forall i, j \in J_n, i < j, \forall r \in H. \quad (29)$$

Nonnegativity: $Y_{jk}, X_{ijk}, Z_k, W_{jr}, B_{ij}^{(r)} \in \{0,1\}; C_{jk}, S_j, F_j, C^{max} \geq 0; U_{jk} \in [0, n]$.

The proposed model integrates machine scheduling, release dates, sequence-dependent setup times, and workforce allocation within a unified framework. The objective function (1) minimizes both the makespan and the total completion time; however, since ($w_1=1$) and ($w_2=0$), only the makespan is optimized. Assignment and chain structure constraints (2–8) ensure that each job is assigned to exactly one machine and that a continuous, cycle-free processing sequence is established on every machine. Timing and setup constraints (9–17, 18–20) define the relationships between start times, completion times, processing times, and sequence-dependent setup times. The makespan definition (21), machine eligibility constraint (22), and MTZ-based sequencing constraints (23–25) guarantee feasible machine assignments and prevent subtours. Release date constraint (26) ensures that no job starts before it becomes available. Workforce constraints (27) enforce the required number of workers for each job, while disjunctive constraints (28–29) prevent a worker from being assigned to overlapping jobs. Finally, the Big-M and planning horizon parameters strengthen the formulation and improve computational efficiency, particularly within the timing (F^*) and workforce (L^*) constraints.

4. Methodology

4.1 The GVNS Metaheuristic

Due to the NP-hardness of the base UPMSP, a General Variable Neighbourhood Search (GVNS) metaheuristic is developed for medium and large instances. The algorithm augments classical VNS with Threshold Acceptance (TA), Adaptive Operator Selection (AOS), Large Neighbourhood Search (LNS) perturbation, Path Relinking (PR) intensification, and a multi-start restart mechanism.

4.2 Solution Representation and Decoding

A solution is a sequence-per-machine representation $\sigma = (\sigma_k)_{k \in K}$. Given sigma, start/finish times and worker assignments are computed by a resource-feasible event-driven decoder: it selects the candidate job with the earliest feasible start, defined as $t_{jk}^{start} = \min\{t: t \geq \max(R_k + s_{i,j,k}, r_j) \text{ and } h_j \text{ worker free at } t\}$ ensuring release-date and worker-capacity feasibility at every step.

4.3 Construction Heuristic

A scored greedy insertion builds the initial solution: jobs are sorted in decreasing minimum-processing-time order (noise $\varepsilon = 0.08$ for diversification). Each job is inserted at the machine-position pair minimizing $Score_{jkl} = \Delta s_{jkl} + \alpha p_{jk} + \beta L_k$ with $\alpha=0.35$ and $\beta=0.15$

4.4 Neighbourhood Structures

Twelve neighbourhood structures N1,...,N12 are employed, partitioned into intra-machine and inter-machine operators (Table 1):

Table 1. Neighbourhood operators used in the GVNS. N11 and N12 target the critical (bottleneck) machine.

ID	Operator	Scope	Description
----	----------	-------	-------------

N1	nbi_insert_within	Intra	Remove one job from position i, reinsert at position j within the same machine sequence.
N2	nbi_swap_two	Intra	Swap two randomly selected jobs within the same machine sequence.
N3	nbi_twoopt_reverse	Intra	Reverse a contiguous sub-sequence [i..j] (2-opt style).
N4	nbi_block_swap	Intra	Swap two non-overlapping blocks within the same machine sequence.
N5	nbi_reverse_all	Intra	Reverse the entire job sequence of a selected machine.
N6	nbx_move_one	Intra	Move one job from machine A to its scored-insertion position on machine B.
N7	nbx_swap_one_one	Intra	Swap one job from machine A with one job from machine B (scored re-insertion).
N8	nbx_swap_two_two	Intra	Swap two jobs from machine A with two jobs from machine B.
N9	nbx_move_two_with_return	Intra	Move two jobs from A to B, then return one job from B to A.
N10	nbx_full_swap_reverse	Intra	Swap the entire sequences of machines A and B; reverse one of them.
N11	nbx_block_relocate	Intra	Move a contiguous block of jobs from the critical machine to another machine.
N12	nbx_block_swap	Intra	Swap a block from the critical machine with a block from the target machine.

4.5 Local Search

The local search cycles through operators {N1-N5, N6, N7, N11} in round-robin fashion, applying up to 20 random trials per operator per cycle. A strict improvement triggers a restart from N1 (first-improvement). Total LS budget: ls_tries = 6000 evaluations per invocation.

4.6 Algorithm Pseudocode

Figure 1 presents the complete GVNS pseudocode as a standalone figure. Key parameters are summarised in Table 2 below ;

Table 2. GVNS parameter settings used in all computational experiments.

Parameter	Value	Description
max_outer	1500	Maximum number of outer VNS iterations.
ls_tries	6000	Maximum local-search evaluations per invocation.
ta_init_frac	0.015	Initial TA threshold as fraction of initial objective.
ta_decay	0.985	Multiplicative decay of TA threshold per iteration.
ta_min_frac	0.001	Minimum TA threshold as fraction of initial objective.
eps (AOS exploration)	0.20	Probability of random operator selection (exploration).
ruin_frac	0.22	Fraction of jobs removed in each LNS ruin step.
lms_period	25	Iterations between periodic LNS perturbation calls.
restart_threshold	300	No improvement iterations before triggering a restart.
max_restarts	3	Maximum number of restarts allowed.
greedy_noise (restart)	0.08	Random noise added to greedy construction at restart.
elite_size	8	Maximum number of elite solutions kept for Path Relinking.
pr_period	30	Iterations between Path Relinking invocations.
pr_steps	10	Number of move steps in each Path Relinking call.

Algorithm 1: GVNS for UPMSP-SDST-RD-HR

INPUT : instance (J, K, p, s, r, h, R) ; params
OUTPUT: best solution x^*

```

1.  $x_0 \leftarrow \text{GreedyScoredInsertion}(J, K, p, s, \text{noise}=0)$ 
2.  $x_{cur} \leftarrow \text{LocalSearch}(x_0, \text{ls\_tries})$ 
3.  $x^* \leftarrow x_{cur}$ 
4.  $\theta \leftarrow \text{obj}(x^*) \times \text{ta\_init\_frac}$  // TA threshold
5. elite =  $\{x^*\}$ ; no_improv = 0; restarts = 0

6. FOR it = 1 TO max_outer DO
7. // — Adaptive Operator Selection —
8. IF rand() <  $\tau$  THEN  $N_k \leftarrow \text{Uniform}(N1..N12)$ 
9. ELSE  $N_k \leftarrow \text{argmax}_i \text{weight}[i]$ 

10. // — Shaking —
11.  $x' \leftarrow N_k(x_{cur})$ 
12. IF it mod lns_period = 0 THEN
13.  $x' \leftarrow \text{LNS\_RuinRecreate}(x_{cur}, \text{ruin\_frac})$ 

14. // — Local Search —
15.  $x'' \leftarrow \text{LocalSearch}(x', \text{ls\_tries})$ 

16. // — Threshold Acceptance —
17. IF  $\text{obj}(x'') < \text{obj}(x_{cur})$  THEN  $x_{cur} \leftarrow x''$ 
18. ELSE IF  $\text{obj}(x'') \leq \text{obj}(x_{cur}) + \theta$  THEN  $x_{cur} \leftarrow x''$ 
19.  $\theta \leftarrow \max(\theta \times \text{ta\_decay}, \theta_{min})$ 

20. // — Global Update —
21. IF  $\text{obj}(x_{cur}) < \text{obj}(x^*)$  THEN
22.  $x^* \leftarrow x_{cur}$ 
23. elite.add( $x^*$ ); no_improv = 0
24.  $\text{weight}[N_k] \leftarrow 0.98 \times \text{weight}[N_k] + 1.0$ 
25. IF  $\text{obj}(x^*) = \text{BestKnown}$  THEN BREAK
26. ELSE
27.  $\text{weight}[N_k] \leftarrow 0.98 \times \text{weight}[N_k]$ ; no_improv++

28. // — Path Relinking —
29. IF it mod pr_period = 0 AND |elite|  $\geq 2$  THEN
30.  $x_{pr} \leftarrow \text{PathRelinking}(x_{cur}, \text{elite}[1], \text{pr\_steps})$ 
31.  $x_{pr} \leftarrow \text{LocalSearch}(x_{pr}, \text{ls\_tries}/3)$ 
32. IF  $\text{obj}(x_{pr}) < \text{obj}(x_{cur})$  THEN  $x_{cur} \leftarrow x_{pr}$ 

33. // — Restart —
34. IF no_improv  $\geq \text{restart\_threshold}$  THEN
35. IF restarts  $\geq \text{max\_restarts}$  THEN BREAK
36.  $x_{cur} \leftarrow \text{GreedyScoredInsertion}(\text{noise} = \text{sg})$ 
37.  $x_{cur} \leftarrow \text{LocalSearch}(x_{cur}, \text{ls\_tries}/2)$ 
38.  $\theta \leftarrow \max(\theta, \text{obj}(x^*) \times \text{ta\_init\_frac})$ 
39. restarts++; no_improv = 0

40. END FOR
41. RETURN  $x^*$ 

```

Figure 1. GVNS pseudocode (Algorithm 1). Steps 8-9: AOS selection. Steps 11-13: shaking / LNS. Steps 17-19: Threshold Acceptance. Steps 21-27: global update and AOS weight update. Steps 29-32: Path Relinking. Steps 34-39: restart

5. Results and Discussion

5.1 Numerical Results

Experiments cover instances from 6 to 100 jobs and 2 to 15 unrelated parallel machines. Small instances ($n \leq 12$) were solved with both the compact MILP (PuLP/Gurobi, TL = 7200 s) and GVNS. Medium ($n=50$) and large ($n=100$) instances were evaluated with GVNS only, as the MILP is intractable at those scales. All experiments ran on a Windows workstation with Intel Core i7 and 16 GB RAM (Table 3).

Table 3. Consolidated computational results. MILP: PuLP/Gurobi compact formulation (TL = 7200 s). GVNS: metaheuristic. '—' = MILP not applied (instance too large). TL = time limit reached; Optimal = proven optimal

Instance	Size	R	MILP Obj.	MILP GAP%	MILP Time(s)	MILP Status	GVNS Obj.	GVNS Time(s)
100 15 S 1-9 6	Large	3	-	-	-	-	604	27.56
100 15 S 1-9 6	Large	5	-	-	-	-	383	25.67
100 15 S 1-9 6	Large	10	-	-	-	-	204	16.66
100 15 S 1-9 6	Large	20	-	-	-	-	112	7.12
100 15 S 1-9 6	Large	40	-	-	-	-	94	4.37
100 15 S 1-9 6	Large	50	-	-	-	-	97	4.86
50 15 S 1-49 2	Medium	3	-	-	-	-	226	10.79
50 15 S 1-49 2	Medium	5	-	-	-	-	166	6.90
50 15 S 1-49 2	Medium	10	-	-	-	-	103	3.42
50 15 S 1-49 2	Medium	20	-	-	-	-	102	1.93
50 15 S 1-49 2	Medium	40	-	-	-	-	100	2.13
50 15 S 1-49 2	Medium	50	-	-	-	-	85	2.41
12 2	Small	3	316	25.9%	7201	TL	384	0.94
12 2	Small	5	-	-	-	-	336	0.69
12 2	Small	10	334	0,00%	3667	Optimal	334	0.68
12 2	Small	20	280	0,00%	4848	Optimal	334	0.68
12 2	Small	40	280	0,00%	6697	Optimal	334	0.7
12 2	Small	50	280	3.6%	7201	TL	334	0.71
12 2	Small	3	316	25.9%	7201	TL	313	0.97
12 2	Small	5	-	-	-	-	280	0.67
12 2	Small	10	334	0,00%	3667	Optimal	280	0.68
12 2	Small	20	280	0,00%	4848	Optimal	280	0.67
12 2	Small	40	280	0,00%	6697	Optimal	280	0.7
12 2	Small	50	280	3.6%	7201	TL	280	0.7
12 20	Small	3	249	29.3%	7200	TL	247	1.64
12 20	Small	5	-	-	-	-	234	0.66
12 20	Small	10	234	51.1%	7200	TL	234	0.66
12 20	Small	20	334	0,00%	6557	Optimal	234	0.69
12 20	Small	40	334	17.4%	7252	Optimal	234	0.71
12 20	Small	50	234	56,00%	8049	Optimal	234	0.73
6 2	Small	3	98	0,00%	0	Optimal	98	1.16
6 2	Small	5	-	-	-	-	79	0.47
6 2	Small	10	79	0,00%	0	Optimal	79	0.47

6_2	Small	20	79	0,00%	0	Optimal	79	0.47
6_2	Small	40	79	0,00%	0	Optimal	79	0.49
6_2	Small	50	79	0,00%	0	Optimal	79	0.49

5.2 Analysis and Discussion

5.2.1 Effect of Worker Capacity R on Makespan

Figure 2 reveals a consistent decreasing-convex relationship between worker capacity R and makespan across all instance sizes. For I_6_2 the objective drops sharply from 98 (R=3) to 79 (R=5) and remains flat thereafter, indicating a saturation threshold at R=5 equal to the peak concurrent demand. The medium instance decreases monotonically from 226 (R=3) to 85 (R=50)—a 62.4% reduction—while the large instance shows the most dramatic response: C_{max} falls from 604 (R=3) to a minimum of 94 (R=40), an 84.4% improvement. These results quantify the economic value of workforce investment: in large-scale contexts, moderate increases in R yield substantial throughput gains, while small instances saturate rapidly.

Figure 1: Effect of Worker Capacity R on Makespan (GVNS)

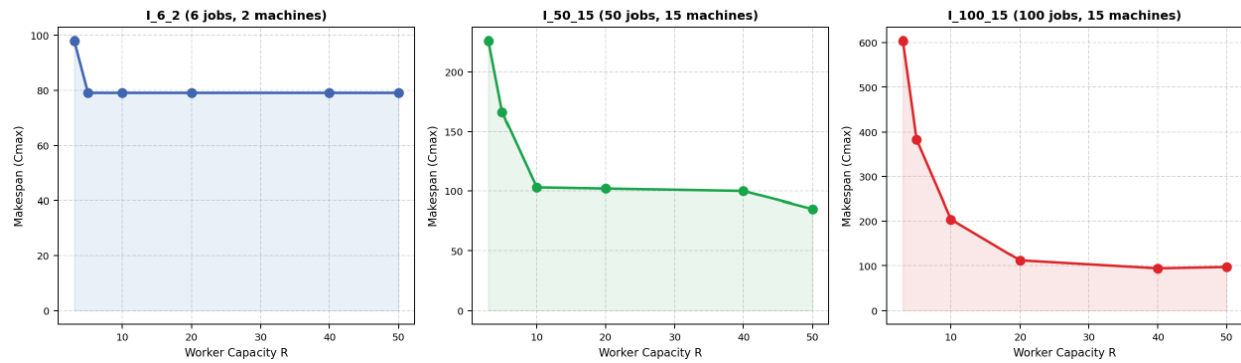


Figure 2. Cmax as a function of worker capacity R for representative instances of each size class (GVNS).

5.2.2 Comparative Analysis Across Small Instances

Figure 3 shows that I_12_2_S_1-9_10 (setup range 1-9) consistently achieves the lowest absolute C_{max} , while I_12_2_S_1-124_4 (setup range 1-124) has the highest, confirming setup magnitude as the dominant factor for 12 job instances. The R=3 to R=5 transition produces the largest marginal improvement; values at R ≥ 10 remain stable. Both GVNS and MILP reach identical objectives wherever the MILP finds provable optimality, validating metaheuristic quality.

Figure 2: GVNS Objective Values for Small Instances Across R Levels

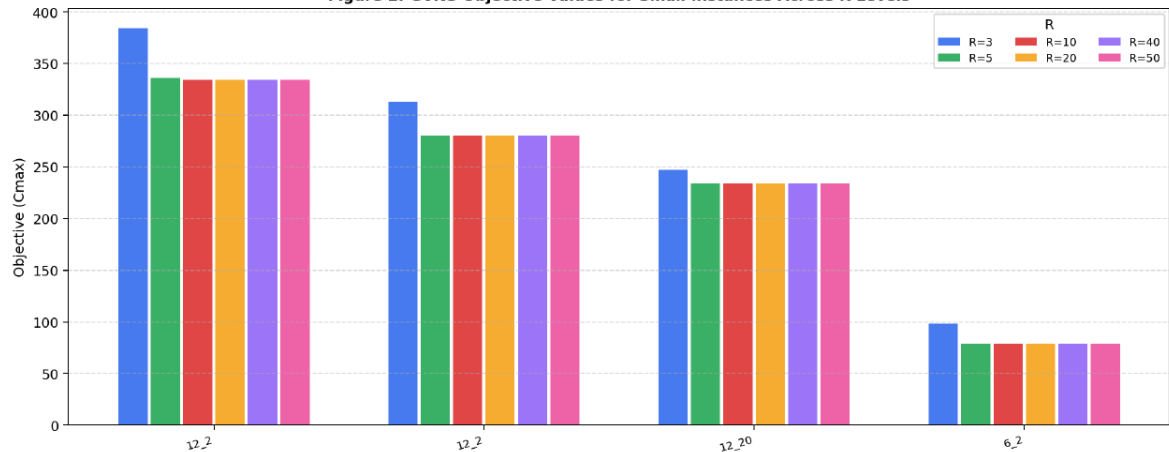


Figure 3. GVNS objective values for all small instances across R in {3,5,10,20,40,50}

5.2.3 Computational Efficiency of GVNS

GVNS runtime decreases as R increases for medium and large instances (Figure 4). At low R many candidate moves produce near-infeasible worker assignments that increase decoding cost; at higher R the constraint is slack and decoding is faster. For the large instance, runtime falls from 27.6 s ($R=3$) to 4.4 s ($R=40$). All GVNS runtimes are orders of magnitude shorter than the MILP (max 28 s vs. 7200 s), confirming the metaheuristic as the only viable option at practical scales.

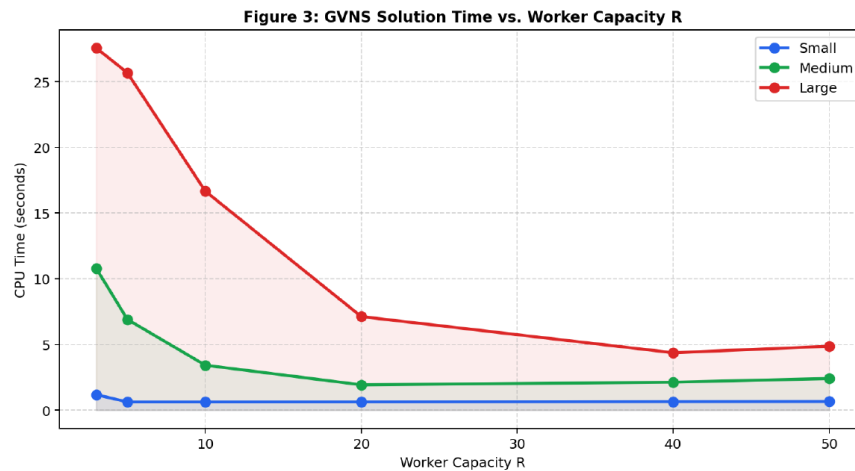


Figure 4. Average GVNS CPU time vs. R , aggregated by instance size class.

5.2.4 MIP Optimality Gap Analysis

MIP gaps range from 0% to above 55%, underscoring computational intractability even at $n=12$. Gaps at $R=3$ are notably higher than at $R \geq 10$ —tighter resource constraints produce more complex branching trees (Figure 5). Proven optimality at $R=20,40$ for some instances shows that relaxed worker constraints tighten the LP relaxation and accelerate branch and-bound. No 12-job instance achieves proven optimality at $R=3$ within 7200 s, confirming the human-resource constraint as the primary driver of MILP difficulty.

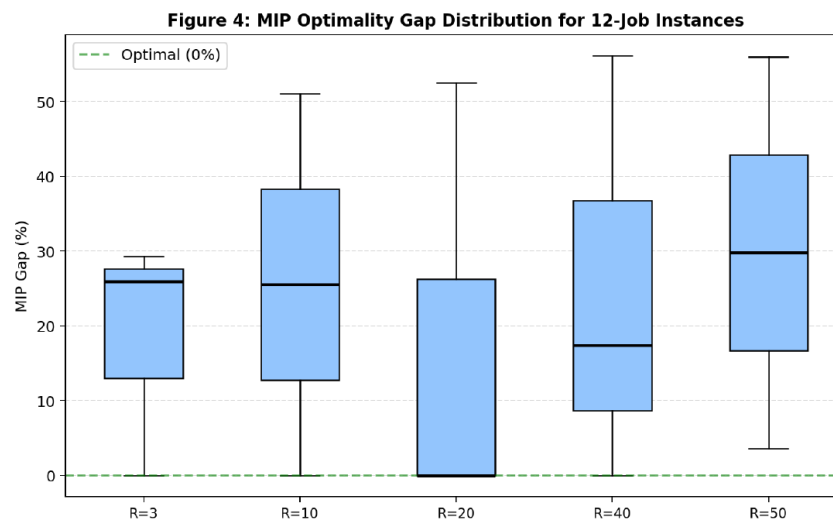


Figure 5. Distribution of MIP optimality gap (%) across 12-job instances.

5.2.5 MILP vs. GVNS Solution Quality

For I_6_2—the only class where MILP reaches proven optimality at all R—both methods produce identical objectives at all six R levels (Figure 6). At R=3 both give obj=98; at R ≥ 5 both give 79. This agreement provides strong empirical validation: whenever a proven optimum is available, GVNS matches it exactly. The GVNS achieves this in under 1.2 s, comparable to the MILP at this scale, while the MILP advantage disappears entirely for larger problems.

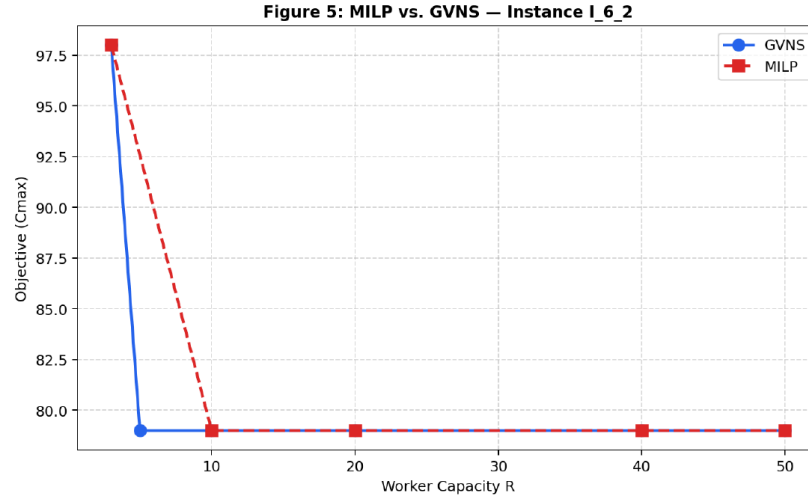


Figure 6. Direct comparison of MILP and GVNS objectives on instance I_6_2.

5.2.6 Large-Scale Behaviour

The large instance shows the steepest improvement between R=10 (obj=204) and R=20 (obj=112) nearly halving C_{max} with 10 additional workers signalling a critical regime transition where most jobs can be processed without queueing for workers (Figure 7). The practical optimum is around R=40 (obj=94, time=4.4 s); the marginal uptick to 97 at R=50 is within metaheuristic stochastic variance.

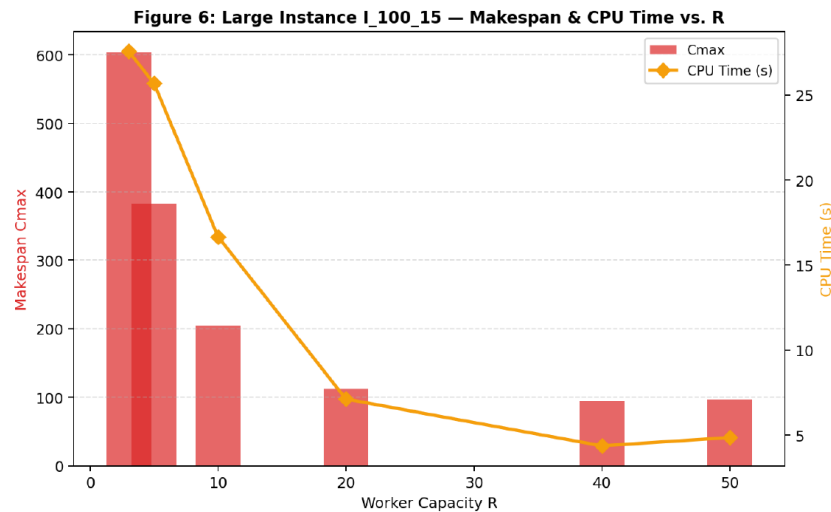


Figure 7. Makespan and CPU time for large instance I_100_15 vs. R.

6. Conclusions

The computational results demonstrate that the proposed GVNS algorithm provides an effective and scalable solution approach for the considered scheduling problem (Table 4). While the compact MILP formulation can solve only small instances optimally (e.g., $n \leq 6$) within negligible computational times, its performance deteriorates significantly for larger instances, often reaching the 7200-second time limit without proving optimality for $n = 12$. In contrast, GVNS successfully solves instances with up to 100 jobs in less than 28 seconds, highlighting its scalability. Regarding

solution quality, GVNS matches the optimal MILP solutions for all $I_{6,2}$ instances and even outperforms the best incumbent MILP solution in the $I_{12,2,S_{1-99},1}$ instance with $R = 3$, obtaining an objective value of 313 compared to 316. The results further reveal that workforce capacity (R) is the primary factor influencing both makespan and computational complexity, with additional workers yielding diminishing returns after a problem-dependent saturation threshold. Finally, the strong performance of GVNS can be attributed to its hybrid design, which combines Adaptive Operator Selection (AOS), Threshold Acceptance, Large Neighborhood Search (LNS) ruin-and-recreate mechanisms, and Path Relinking to achieve an effective balance between exploration and exploitation, while the critical-machine-oriented neighborhood structures (N11 and N12) intensify the search around bottleneck resources, particularly under tight workforce constraints.

Table 4. GVNS vs. MILP for small instances. '= Equal' means GVNS matched MILP objective exactly. Negative values indicate GVNS found a better solution than the MILP incumbent.

Instance	Type	R	GVNS Obj.	GVNS Time(s)	MILP Ref.	Match
12 2	Small	3	384	0.94	316	68
12 2	Small	5	336	0.69	-	-
12 2	Small	10	334	0.68	334	=Equal
12 2	Small	20	334	0.68	280	54
12 2	Small	40	334	0.7	280	54
12 2	Small	50	334	0.71	280	54
12 2	Small	3	313	0.97	316	-3
12 2	Small	5	280	0.67	-	-
12 2	Small	10	280	0.68	334	-54
12 2	Small	20	280	0.67	280	=Equal
12 2	Small	40	280	0.7	280	=Equal
12 2	Small	50	280	0.7	280	=Equal
12 20	Small	3	247	1.64	249	-2
12 20	Small	5	234	0.66	-	-
12 20	Small	10	234	0.66	234	=Equal
12 20	Small	20	234	0.69	334	-100
12 20	Small	40	234	0.71	334	-100
12 20	Small	50	234	0.73	234	=Equal
6 2	Small	3	98	1.16	98	=Equal
6 2	Small	5	79	0.47	-	-
6 2	Small	10	79	0.47	79	=Equal
6 2	Small	20	79	0.47	79	=Equal
6 2	Small	40	79	0.49	79	=Equal
6 2	Small	50	79	0.49	79	=Equal

References

- Akyol Ozer, E. and Sarac, T., MIP models and a matheuristic algorithm for an identical parallel machine scheduling problem under multiple copies of shared resources constraints, *TOP*, vol. 27, pp. 94-124, 2019.
- Al-Harkan, I. M. and Qamhan, A. A., Optimize Unrelated Parallel Machines Scheduling Problems With Multiple Limited Additional Resources, Sequence-Dependent Setup Times and Release Date Constraints, *IEEE Access*, vol. 7, pp. 171533-171547, 2019.
- Avalos-Rosales, O., Angel-Bello, F., and Alvarez, A., Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, *The International Journal of Advanced Manufacturing Technology*, vol. 76, pp. 1705-1718, 2015.

- Báez, S., Angel-Bello, F., Alvarez, A., and Melián-Batista, B., A hybrid metaheuristic algorithm for a parallel machine scheduling problem with dependent setup times, *Computers & Industrial Engineering*, vol. 131, pp. 295–305, 2019.
- Behnamian, J., Zandieh, M., and Fatemi Ghomi, S. M. T., Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm, *Expert Systems with Applications*, vol. 36, no. 6, pp. 9637–9644, 2009.
- Castillo, I., Joro, T., and Li, Y. Y., Workforce scheduling with multiple objectives, *European Journal of Operational Research*, vol. 196, no. 1, pp. 162-170, 2009.
- Cheng, C. Y., Pourhejazy, P., Ying, K. C., Li, S. F., and Chang, C. W., Learning-Based Metaheuristic for Scheduling Unrelated Parallel Machines With Uncertain Setup Times, *IEEE Access*, vol. 8, pp. 74065-74082, 2020.
- Costa, A., Cappadonna, F. A., and Fichera, S., Minimizing the total completion time on a parallel machine system with tool changes, *Computers & Industrial Engineering*, vol. 91, pp. 290–301, 2016.
- Đurasević, M. and Jakobović, D., Heuristic and metaheuristic methods for the parallel unrelated machines scheduling problem: a survey, *Artificial Intelligence Review*, vol. 56, pp. 3181-3289, 2022.
- Ekici, A., Elyasi, M., Özener, O. Ö., and Sarıkaya, M. B., An application of unrelated parallel machine scheduling with sequence-dependent setups at Vestel Electronics, *Computers & Operations Research*, vol. 111, pp. 130–140, 2019.
- Goli, A. and Keshavarz, T., Just-in-time scheduling in identical parallel machine sequence-dependent group scheduling problem, *Journal of Industrial and Management Optimization*, vol. 18, no. 6, pp. 3807-3830, 2022.
- Kim, J. and Kim, H. J., Parallel machine scheduling with multiple processing alternatives and sequence-dependent setup times, *International Journal of Production Research*, vol. 59, no. 18, pp. 5438-5453, 2021.
- Li, Y., Côté, J. F., Coelho, L. C., and Wu, P., Novel efficient formulation and matheuristic for large-sized unrelated parallel machine scheduling with release dates, *International Journal of Production Research*, vol. 60, no. 20, pp. 6104-6123, 2022.
- Lin, Y. K. and Hsieh, F. Y., Unrelated parallel machine scheduling with setup times and ready times, *International Journal of Production Research*, vol. 52, no. 4, pp. 1200-1214, 2014.
- Pei, Z., Wang, G., Jiang, S., Zhou, X., and Zhao, X., A new approximation algorithm for unrelated parallel machine scheduling with release dates, *Annals of Operations Research*, vol. 285, pp. 397-425, 2020.
- Rojanasoonthon, S., Bard, J. F., and Reddy, S. D., Algorithms for parallel machine scheduling: a case study of the tracking and data relay satellite system, *Journal of the Operational Research Society*, vol. 54, no. 8, pp. 812–821, 2003.
- Schutten, J. M. J. and Leussink, R. A. M., Parallel machine scheduling with release dates, due dates and family setup times, *International Journal of Production Economics*, vol. 46-47, pp. 119-125, 1996.
- Thevenin, S., Zufferey, N., and Potvin, J. Y., Makespan minimisation for a parallel machine scheduling problem with preemption and job incompatibility, *International Journal of Production Research*, vol. 55, no. 6, pp. 1588-1606, 2017.
- Wang, L., Wang, S., and Zheng, X., A hybrid estimation of distribution algorithm for unrelated parallel machine scheduling with sequence-dependent setup times, *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 235-246, 2016.
- Yunusoglu, P. and Topaloglu Yildiz, S., Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times, *International Journal of Production Research*, vol. 60, no. 7, pp. 2212-2229, 2022.
- Zhang, L. and Lu, L., Parallel-machine scheduling with release dates and rejection, *4OR*, vol. 14, pp. 165-172, 2016.