informatics

MDPI

Article

Hybrid Flow-Shop Scheduling with Continuous Secondary-Resource Retention: A Feasibility-Preserving Decoder and Simulated Annealing Approach

Nour El Houda Saadani [1], Zied Bahroun [2,*], Mohamed Amine Abdeljaouad [3] and Karam Al-Assaf [2]

[1] Faculty of Sciences of Tunis, University of Tunis El Manar, Tunis 2092, Tunisia; nourelhouda.saadani@isamm.uma.tn
[2] ESM Graduate Program, Department of Industrial Engineering, College of Engineering, American University of Sharjah, Sharjah P.O. Box 26666, United Arab Emirates; g00063499@aus.edu
[3] CESI, CESI LINEACT, 69100 Lyon, France; maabdeljaouad@cesi.fr
* Correspondence: zbahroun@aus.edu

Abstract

Hybrid flow-shop schedules can become infeasible when a scarce secondary resource must remain assigned to a job throughout its complete route rather than only during processing operations. This study examines a hybrid flow-shop problem with continuous renewable-resource retention from first-stage start to final completion, including inter-stage waiting. It develops a mixed-integer linear programming (MILP) formulation of this retention requirement and a deterministic decoder that converts an ordered job sequence into machine assignments, operation timing, and uninterrupted resource reservations. An incumbent-anchored simulated annealing (SA) variant searches the sequence space. The computational study covers 32 configurations: eight exact-validation configurations and 24 larger configurations. The performance measure is the number of planning periods occupied by all jobs. Because this measure equals the transition-count objective plus one period per job, it preserves feasible-schedule rankings, SA objective differences, and the set of MILP-optimal schedules. Across the exact-validation configuration means, SA is 0.41% above the proven optimum on average; across the larger configurations, it improves the constructive heuristic by 3.79%. The results show that the formulation and decoder generate feasible schedules while explicitly accounting for resource occupation during processing and inter-stage waiting.

Keywords: resource-constrained scheduling; renewable resources; occupied-period flow time; healthcare operations; feasibility-preserving decoding; simulated annealing; algorithmic decision support

Check for updates

Academic Editors: Weitian Tong, Lei Chen and Lixin Li

Received: 25 September 2026
Revised: 16 November 2026
Accepted: 24 November 2026
Published: 27 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Resolving scheduling problems is fundamental to improving operational efficiency in manufacturing, healthcare, and service systems [1]. Among these problems, the hybrid flow-shop (HFS) scheduling problem has received sustained attention because it captures production and service environments in which jobs pass through multiple processing stages, with one or more parallel machines available at each stage [2]. By extending classical flow-shop and parallel-machine scheduling, the HFS structure offers greater assignment flexibility but also introduces substantial computational complexity [3,4].

Most HFS research has focused on objectives such as makespan, total flow time, and tardiness under deterministic processing assumptions [5]. In practice, however, machine

availability is rarely the only binding constraint. Many operational systems also require scarce supporting resources, including skilled labor, tools, beds, fixtures, facility space, or other reusable assets that directly affect whether a schedule is feasible [6,7]. Although secondary resources have been considered in several scheduling studies, conventional formulations typically assign these resources at the operation level and allow them to be released between operations [8]. This assumption is not suitable for systems in which a renewable supporting resource must remain assigned to the same job across the full route.

Scheduling models distinguish renewable resources, whose capacity becomes available again after use, from non-renewable resources that are depleted as they are consumed [9,10]. In the present setting, the secondary resource is renewable but subject to a stronger continuity rule: once assigned, it remains occupied from the start of the first operation until final completion. This links waiting time directly to resource availability and strengthens the coupling among sequencing, machine assignment, and secondary-resource use [11,12].

This continuous-allocation pattern appears in multi-stage engineering and service systems. In manufacturing, dedicated fixtures, jigs, molds, tooling, or build plates may remain attached to a job through multiple stages because release and re-mounting can disrupt quality, traceability, or process continuity. In healthcare, scheduling studies similarly show that beds and other supporting capacity affect patient flow and time in the system [13–16]. These examples motivate the continuous secondary-resource requirement studied here: waiting between operations can consume scarce supporting capacity even when no primary machine is processing the job.

This paper studies an HFS setting with R stages, E_r parallel machines at each stage, and L limited renewable secondary resources. Each job must be processed through all stages in sequence while retaining one secondary resource continuously across processing and inter-stage waiting. The objective minimizes total occupied-period flow time, defined as the number of planning periods intersected by each job's route. This period-based measure reflects the planning resolution of the model and gives a positive contribution to a job completed within one period. Lower occupied-period flow time can support reduced work-in-process and more effective use of retained resources, but those operational outcomes are not measured directly in the computational study.

The paper contributes a continuous-retention HFS formulation and a deterministic feasibility-preserving decoder. The formulation keeps the same renewable secondary resource assigned from first-stage start to final completion, including inter-stage waiting, and measures the planning periods during which that capacity remains unavailable. Given an ordered job sequence, the decoder constructs machine assignments, operation timing, and uninterrupted resource reservations. Simulated annealing is used to search the sequence space; the problem-specific contribution lies in the formulation and decoder.

The remainder of this paper is organized as follows. Section 2 reviews the closest literature and positions the continuous-retention problem. Section 3 defines the problem and presents the mathematical model. Section 4 details the deterministic decoder, constructive heuristic, incumbent-anchored simulated annealing variant, and lower bound. Section 5 presents the computational experiments and results. Section 6 concludes and outlines future research.

2. Literature Review

Flow-shop and hybrid flow-shop scheduling have a long research history. Ribas et al. [17] classify hybrid flow shops from production-system and solution-procedure perspectives, while Çolak and Keskin [4] provide a broad systematic review of more recent HFS research. Neufeld et al. [18] show how multi-objective formulations have expanded the field beyond single-objective makespan minimization, and Utama et al. [19] document

the growing attention to energy-efficient variants. These reviews establish the breadth of the field but also show that comparability depends on the exact shop structure, resource semantics, and performance measure.

This review is organized to make the research state and this paper's position explicit. Section 2.1 summarizes the dominant and alternative objectives in an HFS. Section 2.2 focuses on HFS formulations that incorporate secondary resources beyond machines. Section 2.3 reviews solution approaches for a resource-constrained HFS, emphasizing metaheuristics used when exact models do not scale. Finally, Section 2.4 summarizes recent learning-based and transfer optimization approaches and clarifies why their feasibility structure differs from the setting studied here.

2.1. Optimization Objectives in HFS Scheduling

Makespan remains a dominant system-level objective. Wang et al. [20] develop an iterated-greedy approach for a cascaded flow-shop setting, while Rahmativala and Ghahremani [21] address uncertain setup times through robust optimization. Fu et al. [22] combine processing quality and tardiness in a stochastic HFS model. These studies illustrate the diversity of objectives and uncertainty structures, but they do not impose continuous retention of one renewable secondary resource through waiting.

Makespan alone does not capture how long individual jobs remain in the system. Öztop et al. [23] study total flow-time minimization in hybrid flow shops, while Jerbi et al. [24] jointly consider makespan, patient flow time, and doctors' workloads in a healthcare setting. Both studies confirm the relevance of flow-oriented objectives, but neither imposes the same-resource full-route retention rule considered here.

The literature also shows that realistic scheduling requires objectives and feasibility conditions to be designed together. Application-oriented reviews [18,25] and healthcare planning studies [26,27] show that service objectives interact with scarce resources and uncertainty. The present study therefore aligns its period-based performance measure with the operational interval during which the secondary resource remains occupied.

Occupied-period flow time is therefore a meaningful primary objective when waiting and retained-resource use are operationally important. It also requires sequencing and machine decisions to be coordinated with the release state of supporting resources rather than treating performance and feasibility as separate design choices.

2.2. Secondary Renewable Resources in Hybrid Flow-Shop Scheduling

Secondary-resource formulations vary substantially. Costa et al. [8] model limited human resources in a hybrid flow shop. Figielska [12,28] studies shared renewable resources in two-stage flow-shop environments, and Mlekusch and Hartl [29] analyze a dual-resource-constrained re-entrant flexible flow shop. These studies establish that secondary resources materially alter feasibility and performance, but their resources are assigned at operation level or follow release semantics different from full-route retention.

Healthcare studies reinforce the operational importance of supporting capacity. Bed-occupancy scheduling [15], hospital logistics [16], and multi-resource surgical planning [30] show that non-machine resources can determine delay and throughput. These studies motivate the bed-like interpretation used here, but they do not provide a directly comparable HFS benchmark with the same continuous-retention rule and occupied-period objective.

2.3. Metaheuristics for Resource-Constrained Hybrid Flow-Shop Scheduling

Metaheuristics are widely used when exact scheduling models lose tractability. Hajji et al. [31] apply simulated annealing to a hybrid flow-shop setting. Mlekusch and Hartl [29] combine constraint programming with a hybrid genetic algorithm, while Tao et al. use

imperialist-competitive [32] and self-adaptive artificial-bee-colony [33] search in resource-constrained hybrid flow shops. For the problem considered here, the essential design element is the decoder that maps a job permutation to a schedule while enforcing precedence, machine capacity, secondary-resource exclusivity, and uninterrupted retention.

Related parallel-machine and flexible-flow-shop studies also use exact, heuristic, and metaheuristic methods for shared or additional resources [9,34–36]. Their results cannot be transferred numerically to the present model because the shop structure, objective, or resource-release rule differs. They nevertheless reinforce the importance of problem-specific feasibility handling.

Across the closest resource-constrained HFS studies, secondary resources are generally required during processing and may be released between operations [8,12,28,36]. Flow-time-oriented HFS studies, in turn, do not impose continuous retention of the same supporting resource during inter-stage waiting [23,24]. The present formulation brings these two dimensions together.

2.4. Learning-Based and Transfer Optimization Approaches for Scheduling

Recent scheduling research also explores data-driven and learning-based control. Wang et al. [37] develop a real-time reinforcement-learning framework for HFS scheduling. Chen et al. [38] use a Transformer pointer network for transfer optimization, Zhou et al. [39] study knowledge transfer across scheduling tasks, Xie et al. [40] combine transfer learning with multi-objective evolutionary search, and Zhang et al. [41] develop multitask genetic-programming hyperheuristics.

These approaches can accelerate repeated scheduling decisions when suitable state representations and training data are available. The present work instead uses a transparent deterministic decoder to handle the long-horizon coupling between machine timing and retained resources. Learning-based control is not examined.

For the continuous-retention setting, any learning or search method must preserve one secondary-resource assignment over the complete job route, including inter-stage waiting. The decoder developed here handles that coupling directly by reconstructing machine assignments, timing, and resource reservations from each job sequence before the candidate is evaluated.

Table 1 compares the present study with the closest literature along four dimensions: shop structure, secondary-resource semantics, objective, and retention during waiting. Prior studies establish the foundations for renewable-resource HFS scheduling and flow-time optimization. The present formulation combines total occupied-period flow-time minimization with continuous retention of the same renewable secondary resource throughout processing and inter-stage waiting.

Table 1. Summary of representative studies on hybrid flow-shop scheduling.

Authors	Objective(s)	Type of Resource Considered	Method(s)	Relation to the Present Problem
[12]	Makespan	Machines + renewable shared resources	Heuristic approach	HFS with renewable shared resources, but allocation is operation-level and resources may be released between operations; no continuous retention through waiting.

Table 1. *Cont.*

Authors	Objective(s)	Type of Resource Considered	Method(s)	Relation to the Present Problem
[8]	Makespan	Machines + human resources (renewable)	Mixed Integer Programming	HFS with renewable human resources and a makespan objective; resource assignment is operation-level rather than retained over the full job route.
[7]	Makespan, worker fatigue	Machines + workforce (renewable)	Agent-based simulation + optimization	Adds workforce and fatigue realism, but does not impose continuous renewable-resource retention over the full route.
[24]	Makespan + flow time + workload balance	Machines + doctors (renewable)	Genetic Algorithm + dispatching rules	Includes flow time and renewable doctors, but not continuous resource retention during inter-stage waiting.
[42]	Integrated production + distribution	Machines + distribution resources	Memetic computing	Integrates production and distribution resources, whose semantics do not create the same full-route renewable-resource coupling.
[35]	Makespan with setup times	Machines + additional resources	GRASP heuristic	Uses additional resources in a related parallel-machine setting, but with a different shop structure and resource-retention rule.
[32]	Makespan + energy	Machines + energy	Discrete Imperialist Competitive Algorithm	Resource-aware flow shop with energy objectives; it does not combine total flow time with continuous renewable-resource retention.
Present study	Total occupied-period flow time	Machines + continuous renewable resource	MILP + deterministic decoder + incumbent-anchored SA search	Combines occupied-period flow-time minimization with continuous retention of the same renewable secondary resource through processing and inter-stage waiting.

Hybrid flow-shop scheduling is well known to be computationally difficult. In particular, Gupta [43] proves NP-hardness already for a two-stage hybrid flow shop with machine multiplicities (2,1), and Hoogeveen et al. [44] provide further NP-hardness results for closely related two-stage flow-shop variants with parallel processing structures. These established results indicate that even baseline HFS formulations become challenging as instance size grows. In the setting studied here, continuous secondary-resource retention further couples waiting, sequencing, and resource availability, which motivates the use of exact optimization for validation and heuristic search for larger instances.

The closest related studies either model renewable secondary resources in hybrid flow shops but treat them as operation-level requirements that can be released between stages [8,12], or consider flow-time-oriented performance without the same continuous supporting-resource requirement [23,24]. This leaves a clear gap for settings in which one renewable secondary resource must remain assigned continuously to each job across

processing and inter-stage waiting. The formulation and decoder developed below address that coupling explicitly.

Among the closest studies reviewed, we did not identify a published benchmark combining the same HFS structure, continuous retention of one secondary resource throughout processing and waiting, the same-period operation rule, and the occupied-period objective. Objective values from adjacent studies are therefore not directly comparable. The evaluation uses proven MILP optima on tractable configurations, the analytical lower bound, and the constructive decoder as formulation-consistent references. These references assess exact solution quality, bound tightness, and improvement over the constructive sequence. Same-decoder comparisons with alternative search procedures are outlined in Section 6.

3. Problem Definition and Mathematical Model

This section defines the scheduling environment and formulates the corresponding MILP. The model first represents machine assignment, processing, precedence, and timing, and then extends secondary-resource occupation across the complete interval from first-stage start to final completion.

3.1. Problem Description

This paper studies a resource-constrained hybrid flow shop motivated by hospital patient-flow scheduling. Jobs represent patients moving through successive procedure types (stages), and machines represent identical parallel equipment units available at each stage, such as imaging devices, laboratories, or treatment rooms. In addition to machine capacity, each job requires one unit of a secondary renewable resource drawn from a limited pool of identical resources, such as a hospital bed. The bed-like resource remains assigned from the start of the first procedure until completion of the last, including inter-stage waiting, which creates the continuous-retention constraint modeled below.

A set of N independent jobs must be processed through R stages in the same fixed order. At each stage r, there are E_r identical parallel machines, and each job requires exactly one machine at each stage. Processing is non-preemptive. Each job's operation at stage $r + 1$ can start only after its operation at stage r has completed (precedence constraints). At any time, each machine can process at most one job.

The secondary resource constraint is the key feature. Each job must be assigned exactly one secondary resource from the available pool. That assigned resource must remain continuously allocated to the job from the start of the first-stage operation until the completion of the last-stage operation, including all waiting times between stages. Therefore, once a job starts, its resource is locked to it and cannot be used by any other job until the job finishes its final stage. At any time, each resource can be assigned to at most one job. This models settings such as inpatient pathways where a patient occupies the same bed throughout the entire stay, or manufacturing settings where a fixture or mold remains dedicated until completion.

Time is represented at two linked resolutions, each with a distinct role. Sub-periods are the basic scheduling units for processing durations, precedence, and machine and resource capacity. Planning periods represent operational service blocks, such as days or shifts, and define both the same-period completion rule and the reporting unit for occupied-period flow time. An equivalent formulation could use a single linear sub-period index together with a derived period-membership mapping. The two-level representation is retained because it mirrors the operational input structure, makes the same-period completion rule explicit, and reports occupied-period resource use directly. All computational experiments require each operation to start and finish within one planning period; operations that cross planning-period boundaries are not considered.

The objective minimizes total occupied-period flow time. For each job, the measure counts the planning periods intersected from first-stage start to final completion. Because the same secondary resource remains assigned over that interval, the objective also measures the planning periods during which this capacity is unavailable to other jobs. It is therefore aligned directly with the defining resource constraint. Exact elapsed sub-period flow time, machine utilization, and inter-stage waiting are complementary measures rather than components of the objective.

Figure 1 shows the hybrid flow-shop structure with parallel machines at each stage and a secondary renewable resource retained over the complete job route. The resource remains occupied during both processing and inter-stage waiting.

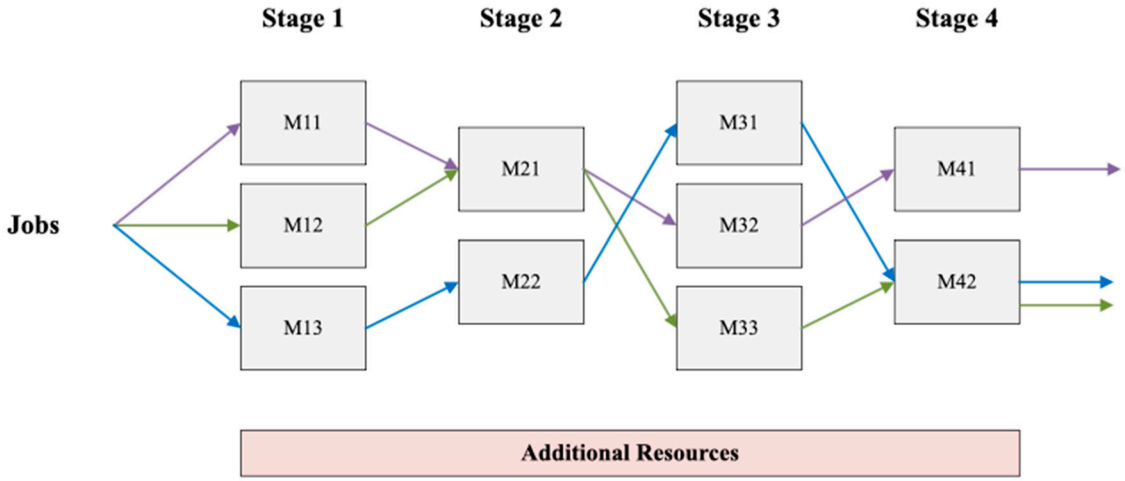

Figure 1. Hybrid flow-shop structure with parallel machines and continuous secondary-resource retention across processing and waiting periods.

3.2. Notation and Complexity

Following the three-field notation framework of Graham et al. [45], standard scheduling descriptors identify the shop environment and objective. The scheduling literature does not provide a standard qualifier for continuous full-route retention of the same renewable secondary resource. Sections 3.1 and 3.3 therefore define the HFS environment, retention rule, and occupied-period objective explicitly rather than introducing a manuscript-specific three-field code.

This explicit description is used throughout the mathematical model and computational study.

Hybrid flow-shop scheduling problems are known to be NP-hard when the number of stages is at least two. This result holds for classical objectives such as makespan and remains relevant for occupied-period flow-time minimization. Continuous renewable-resource retention adds assignment and synchronization constraints that further couple waiting, sequencing, and capacity. These factors justify exact optimization for validation and heuristic search for larger instances.

3.3. Mathematical Formulation

This section introduces the notation and decision variables and then presents the active mathematical formulation used in the computational study.

The parameters used in this formulation are explained in detail below.

- N, i: number of jobs and job index, respectively;
- R, r: number of stages and stage index, respectively;
- E_r: number of identical parallel machines available at stage r;

- e: machine index;
- T, t: number of planning periods and period index, respectively;
- L, l: number of secondary renewable resources and resource index, respectively;
- C, c: number of sub-periods per period and sub-period index, respectively;
- $p_{i,r}$: processing duration of job i at stage r;
- M: sufficiently large positive constant used in the big-M constraints;
- ε: small positive boundary offset; $\varepsilon = 0.001$ in the solved model.

The large constant M is used only in the big-M constraints. The maximum timestamp in the computational design is $25 \times 32 = 800$ sub-periods; therefore, $M = 1000$ safely dominates every feasible timestamp difference. The value $\varepsilon = 0.001$ is smaller than $1/C$ for both planning resolutions used in the study ($C = 24$ and $C = 32$), which resolves planning-period boundary ambiguity without excluding any integer feasible solution.

The decision variables that define the optimization model are presented below.

- $X_{i,l,t,c}$: 1 if secondary resource l is occupied by job i in sub-period c of period t, and 0 otherwise;
- $Y_{i,r,e,t,c}$: 1 if job i occupies machine e at stage r in sub-period c of period t, and 0 otherwise;
- $U_{i,r,e}$: 1 if job i at stage r is assigned to machine e, and 0 otherwise;
- $Z_{i,l}$: 1 if job i is assigned secondary resource l, and 0 otherwise;
- $S_{i,r}$: start time, in sub-period units, of job i at stage r;
- $C_{i,r}$: completion time, in sub-period units, of job i at stage r;
- S_i: job-level start time expressed in period units;
- C_i: job-level completion time expressed in period units.

The formulation uses standard scheduling symbols with comma-separated subscripts. Operation-level start and completion times are denoted $S_{i,r}$ and $C_{i,r}$; job-level start and completion periods are S_i and C_i. The occupancy and assignment variables use only their applicable indices. The additional variables $D_{i,r}$, A_i, and B_i, introduced in Section 3.5, are integer period or timestamp variables.

$$Min \sum_{i=1}^{N} (C_i - S_i + 1) \tag{1}$$

$$\sum_{l=1}^{L} Z_{i,l} = 1 \qquad \forall i = 1, \ldots N \tag{2}$$

$$\sum_{i=1}^{N} X_{i,l,t,c} \leq 1 \qquad \forall l = 1, \ldots L;\ \forall t = 1, \ldots T;\ \forall c = 1, \ldots C \tag{3}$$

$$X_{i,l,t,c} \leq Z_{i,l} \qquad \forall i = 1, \ldots N;\ \forall l = 1, \ldots L;\ \forall t = 1, \ldots T;\ \forall c = 1, \ldots C \tag{4}$$

$$\sum_{i=1}^{N} Y_{i,r,e,t,c} \leq 1 \quad \forall r = 1, \ldots R; \forall e = 1, \ldots E_r;\ \forall t = 1, \ldots T;\ \forall c = 1, \ldots C \tag{5}$$

$$\sum_{e=1}^{E_r} \sum_{t=1}^{T} \sum_{c=1}^{C} Y_{i,r,e,t,c} = p_{i,r} \qquad \forall i = 1, \ldots N;\ \forall r = 1, \ldots .R \tag{6}$$

$$\sum_{e=1}^{E_r} U_{i,r,e} = 1 \qquad \forall i = 1, \ldots N;\ \forall r = 1, \ldots .R \tag{7}$$

$$Y_{i,r,e,t,c} \leq U_{i,r,e} \quad \forall i = 1, \ldots N;\ \forall r = 1, \ldots R;\ \ \forall e = 1, \ldots E_r;\ \forall t = 1, \ldots T;\ \forall c = 1, \ldots C \tag{8}$$

$$\sum_{i=1}^{N} \sum_{c=1}^{C} Y_{i,r,e,t,c} \leq C \qquad \forall r = 1, \ldots R; \forall e = 1, \ldots E_r; \forall t = 1, \ldots T \tag{9}$$

$$S_{i,r} \leq \sum_{e=1}^{E_r} Y_{i,r,e,t,c}[c + C(t-1)] + M\left(1 - \sum_{e=1}^{E_r} Y_{i,r,e,t,c}\right)$$

$$\forall i = 1, \ldots, N;\ \forall r = 1, \ldots, R;\ \forall t = 1, \ldots, T;\ \forall c = 1, \ldots, C. \tag{10}$$

$$C_{i,r} \geq \sum_{e=1}^{E_r} Y_{i,r,e,t,c}[c + C(t-1)]$$

$$\forall i = 1, \ldots, N;\ \forall r = 1, \ldots, R;\ \forall t = 1, \ldots, T;\ \forall c = 1, \ldots, C. \tag{11}$$

$$S_{i,r} \geq 1; \quad C_{i,r} \geq 1 \qquad \forall i = 1, \ldots N;\ \forall r = 1, \ldots R \tag{12}$$

$$C_{i,r} - S_{i,r} = p_{i,r} - 1 \qquad \forall i = 1, \ldots N;\ \forall r = 1, \ldots R \tag{13}$$

$$S_{i,r+1} \geq C_{i,r} + 1 \qquad \forall i = 1, \ldots N;\ \forall r = 1, \ldots R - 1 \tag{14}$$

$$\sum_{r=1}^{R} \sum_{e=1}^{E_r} Y_{i,r,e,t,c} = \sum_{l=1}^{L} X_{i,l,t,c} \qquad \forall i = 1, \ldots N;\ \forall t = 1, \ldots T;\ \forall c = 1, \ldots C \tag{15}$$

$$\frac{S_{i,1}}{C} \leq S_i \leq \frac{S_{i,1}}{C} + 1 - \varepsilon \qquad \forall i = 1, \ldots N \tag{16}$$

$$\frac{C_{i,R}}{C} \leq C_i \leq \frac{C_{i,R}}{C} + 1 - \varepsilon \qquad \forall i = 1, \ldots N \tag{17}$$

$$S_i \geq 1; \quad C_i \geq 1 \qquad \forall i = 1, \ldots N \tag{18}$$

$$X_{i,l,t,c},\ Y_{i,r,e,t,c},\ U_{i,r,e},\ Z_{i,l} \in \{0, 1\} \tag{19}$$

$$S_{i,r},\ C_{i,r},\ A_i,\ B_i \in \{1, \ldots, \mathrm{TC}\} \tag{20}$$

$$S_i,\ C_i,\ D_{i,r} \in \{1, \ldots, \mathrm{T}\} \tag{21}$$

3.4. Interpretation of the Objective Function and Constraints

Objective function (1) minimizes total occupied-period flow time. For each job, $C_i - S_i + 1$ counts the planning periods intersected by the route from first-stage start to final completion. A job that starts and finishes within the same period therefore contributes one occupied period rather than zero. This is a period-index measure, not exact elapsed sub-period time, and the terms "occupied-period flow time," "occupied-period span," and "resource-occupation interval" are used consistently in the remainder of the paper.

Equation (2) ensures that each job utilizes only one additional resource. Inequality (3) establishes that each resource can be occupied by at most one job at any given sub-period. Inequality (4) ensures that a job can occupy a secondary resource only if that resource is assigned to the job.

Constraint (5) ensures that each machine can accommodate at most one job per sub-period. Equation (6) guarantees that each job is processed exactly once at each required stage, with the operation spanning a number of sub-periods equal to its duration. Equation (7) ensures that each job i, requiring an operation at stage r, is assigned to a single machine at that stage. Inequality (8) ensures that at any sub-period, each operation of job i at a specific stage exclusively occupies the assigned machine.

Constraint (9) limits the operating time of each machine per period, ensuring it does not exceed the available sub-periods in that period. Constraints (10)–(12) determine the start and end times of each job's operation at a given stage, measured in sub-period units. Equation (13) enforces that the sub-periods allocated for processing an operation on a machine are consecutive.

Inequality (14) establishes stage precedence. Constraint (15) is the processing-only linkage between machine processing and resource occupation.

Constraints (16)–(18) define the job-level start and completion periods. Constraints (16) and (17) map the first-stage start and final-stage completion from sub-period timestamps to planning-period indices, while Constraint (18) imposes positive lower bounds. Constraints (19)–(21) state the binary and integer domains explicitly. The OPL implementation uses Boolean declarations for X, Y, U, and Z and integer declarations for all time, period-index, and resource-occupation endpoint variables.

3.5. Same-Period Operation Completion and Continuous Resource Retention

The formulation used in the computational study requires every operation to complete within the planning period in which it starts. This restriction provides an unambiguous mapping from sub-period timestamps to planning-period indices. The following variable identifies the planning period containing the start of each operation:

- $D_{i,r}$: period containing the start of job i at stage r.

The active same-period formulation uses the following constraints:

$$\frac{S_{i,r}}{C} \leq D_{i,r} \leq \frac{S_{i,r}}{C} + 1 - \varepsilon \qquad \forall i = 1, \ldots N;\ \forall r = 1, \ldots R \tag{22}$$

$$D_{i,r} \geq 1 \qquad \forall i = 1, \ldots N;\ \forall r = 1, \ldots R \tag{23}$$

$$S_{i,r} + p_{i,r} - 1 \leq D_{i,r}\, C \qquad \forall i = 1, \ldots N;\ \forall r = 1, \ldots R \tag{24}$$

Inequalities (22) and (23) determine the planning period containing the start of each operation, while Constraint (24) enforces completion within that same period. The computational analysis therefore concerns the same-period formulation; operations that cross planning-period boundaries are not modeled.

The processing constraints identify machine occupation, while the continuous-allocation setting additionally requires the assigned secondary resource to remain reserved during both processing and inter-stage waiting. The following variables and constraints enforce uninterrupted resource retention from the start of the first operation to completion of the final operation:

- A_i: start, in sub-period units, of continuous secondary-resource occupation by job i.
- B_i: completion, in sub-period units, of continuous secondary-resource occupation by job i.

For the continuous-retention formulation used in the computational study, processing-only Constraint (15) is replaced by Inequality (25), and Constraints (26)–(30) are added:

$$Y_{i,r,e,t,c} \leq \sum_{l=1}^{L} X_{i,l,t,c} \qquad \forall i = 1, \ldots N\ ;\ \forall r = 1, \ldots R;\ \forall e = 1, \ldots E_r;\ \forall t = 1, \ldots T;\ \forall c = 1, \ldots C \tag{25}$$

$$\sum_{t=1}^{T} \sum_{c=1}^{C} \sum_{l=1}^{L} X_{i,l,t,c} = C_{i,R} - S_{i,1} + 1 \quad \forall i = 1, \ldots N \tag{26}$$

$$A_i \leq \sum_{l=1}^{L} \left(X_{i,l,t,c} * (c + C \times (t-1)\ ) \right) + M\left(1 - \sum_{l=1}^{L} X_{i,l,t,c}\right) \tag{27}$$

$$B_i \geq \sum_{l=1}^{L} \left(X_{i,l,t,c} \times (c + C * (t-1)\ ) \right) \tag{28}$$

$$A_i \geq S_{i,1} \qquad \forall i = 1, \ldots N; \tag{29}$$

$$B_i \leq C_{i,R} \qquad \forall i = 1, \ldots N; \tag{30}$$

Inequality (25) forces the assigned resource to be occupied whenever the job is being processed. Constraints (27) and (28) apply for all jobs, planning periods, and sub-periods; their repeated index qualifications are stated here rather than appended to the long-displayed equations. Constraint (26) makes the total number of occupied resource sub-periods equal to the complete span from the first-stage start to final-stage completion. Constraints (27) and (28) bind the occupied timestamps to the resource-occupation endpoints, and Constraints (29) and (30) place those endpoints inside the job's start-completion interval. Together with the at-most-one-resource-per-sub-period constraints, these relations leave no gap inside the job span and therefore enforce uninterrupted secondary-resource retention during both processing and waiting.

The formulation therefore distinguishes primary processing capacity from retained secondary-resource availability. A schedule is feasible only when precedence, machine capacity, single-resource assignment, resource exclusivity, and continuous retention are satisfied simultaneously.

4. Solution Approach

The solution framework combines exact optimization with decoder-based heuristic methods. The MILP is solved with IBM ILOG CPLEX 12.5.1 to obtain proven optima for tractable instances. These exact solutions provide the reference set used to evaluate the lower bound, constructive heuristic, and SA.

For larger instances, a constructive heuristic generates an initial feasible schedule and SA searches for improved job sequences. Both procedures use the same feasibility-preserving decoder and minimize total occupied-period flow time from first-stage start to final completion.

4.1. Solution Representation and Encoding

A candidate schedule is represented by an ordered list of jobs, denoted LS. The sequence specifies only the order in which jobs are presented to the decoder. Given LS, the deterministic decoder selects one renewable secondary resource for each job's complete route and constructs the machine assignments, start times, completion times, and uninterrupted resource reservation. The assigned secondary resource remains fixed from the first-operation start to final completion, including all inter-stage waiting. SA modifies only the job order; every remaining scheduling decision is reconstructed by the decoder.

In this illustrative two-stage example (Table 2), each stage has one machine, two secondary resources are available, and each planning period contains eight sub-periods. The processing-time vectors are $p(J2) = (2, 3)$, $p(J1) = (3, 2)$, and $p(J3) = (2, 2)$. Jobs J2 and J1 occupy one planning period. Job J3 finishes Stage 1 at sub-period 7, but its Stage 2 operation is placed at sub-periods 9–10 because an operation may not cross the period boundary. Resource R1 therefore remains assigned from sub-period 6 through 10, including the inter-stage waiting interval, and J3 contributes two occupied planning periods. The example explains the encoding and decoding logic and is not part of the computational dataset.

Table 2. Illustrative decoding of the ordered sequence $LS = [J_2, J_1, J_3]$.

Sequence Position and Job	Assigned Resource	Stage 1 Assignment	Stage 2 Assignment	Resource Occupation	Occupied Periods
1: J_2	R_1	M_1, 1–2	M_2, 3–5	1–5	1
2: J_1	R_2	M_1, 3–5	M_2, 6–7	3–7	1
3: J_3	R_1	M_1, 6–7	M_2, 9–10	6–10, including waiting	2

Proposition 1. *Feasibility preservation of returned schedules. Whenever the decoder returns a schedule for an input job sequence, that schedule satisfies stage precedence, machine capacity, single-resource assignment, resource exclusivity, same-period operation completion, and continuous secondary-resource retention.*

Proof. Consider any job scheduled by the decoder. Operations are placed in technological order, and the operation at stage r + 1 starts no earlier than completion of the operation at stage r; therefore, stage precedence is satisfied. At each stage, an operation is assigned

only to a free machine-time interval, and that interval is removed from the machine calendar after commitment. Consequently, two operations cannot occupy the same machine simultaneously. □

Before the job is scheduled, the decoder selects one available secondary resource. That resource remains assigned for the complete route and cannot be released during inter-stage waiting. The first-stage operation cannot start before the resource becomes available, and the resource remains unavailable to subsequent jobs until final-stage completion. Resource exclusivity and uninterrupted retention are therefore preserved.

If a required operation cannot be placed within the planning horizon, the decoder rejects the construction rather than returning it. Therefore, every schedule returned by the decoder satisfies the modeled feasibility conditions.

Proposition 1 establishes feasibility preservation, not decoder completeness. A sequence rejected by the deterministic placement policy may admit a feasible schedule under different machine assignments, resource choices, or tie-breaking decisions. The code selects the resource with the earliest release time and retains the first eligible resource when release times tie. For machine placement, it uses a deterministic scan over machine and free-interval candidates; equal-start alternatives do not replace the retained candidate, while eligible earlier starts encountered by the scan can do so. One permutation therefore maps to at most one deterministic decoder trajectory, which restricts the portion of the full feasible schedule space explored by SA.

4.2. Constructive Heuristic for the Initial Solution

In the constructive heuristic, the decoder builds a schedule from LS by assigning one secondary resource to each job and reserving it continuously, then placing operations through its deterministic machine-and-interval scan. The heuristic first computes the theoretical minimum occupied-period span for each job using Equation (31).

$$\text{MinStay}_i = \left\lceil \frac{\sum_{r=1}^{R} p_{i,r}}{C} \right\rceil \tag{31}$$

Processing times are indexed by job and stage, and C denotes the number of sub-periods per planning period. Equation (31) gives the ceiling of the sequential processing workload divided by C, which is the corresponding job-level lower bound under the occupied-period convention.

The primary variables used in this approach include:

- $Res[l]$: earliest sub-period at which secondary resource l becomes available.
- $Machine[r,e,t,c]$: machine-availability array; 0 indicates that machine e at stage r is available in sub-period c of period t, while a nonzero value indicates occupancy.
- $Round[i]$: assignment round for job i.
- $start[i]$: starting period of job i.
- $finish[i]$: finishing period of job i.

Initially, all entries of the resource-availability vector Res[l] and machine-availability array Machine[r,e,t,c] are set to zero, while Round[i] is set to 1. For the initial constructive solution, jobs are sorted in decreasing total processing time; for SA candidates, the input order is preserved. In Round 1, the decoder commits a job only when the resource and machine windows selected by its deterministic scan allow the theoretical minimum occupied-period span to be attained. This is a packing priority, not a requirement that every feasible job must attain the lower bound.

In Round 2, jobs deferred from Round 1 are placed using the same deterministic scan even when contention adds occupied periods. If a Round-2 placement fails within

the horizon, the implemented procedure resets the calendars and retries the complete job sequence with the minimum-span restriction disabled. The candidate is returned as infeasible only if this fallback construction also fails. Different tie-breaking or assignment choices could still produce another feasible schedule, consistent with the completeness limitation stated above.

Algorithm 1 summarizes the constructive heuristic and the deterministic feasibility-preserving decoder.

Algorithm 1. Constructive heuristic and deterministic decoder.

Input: LS, N, L, R, E_r, $p_{i,r}$, T, and C.
1. If LS is the initial sequence, sort jobs by decreasing total processing time; otherwise retain the candidate order.
2. Initialize resource release times, machine calendars, and all job-assignment indicators.
3. For Round = 1, 2:
4. For each unscheduled job in LS:
5. Select the secondary resource with the smallest release time; retain the first indexed resource when tied.
6. Construct the route stage by stage using the deterministic machine-and-interval scan.
7. In Round 1, commit the route only if its occupied-period span attains Equation (31); otherwise defer the job.
8. In Round 2, commit the feasible route even when contention adds occupied periods.
9. If the two-round construction fails, reset the calendars and retry the complete order with the minimum-span restriction disabled.
10. Return the feasible schedule and objective, or return infeasible if the fallback construction also fails.

4.3. Incumbent-Anchored Simulated Annealing Variant

The study uses an incumbent-anchored SA variant rather than conventional current-state simulated annealing. It maintains a working sequence and a separately stored feasible incumbent. Each candidate is generated by swapping two independently sampled positions in the working sequence and decoding the complete order. A feasible candidate with an objective no worse than the incumbent remains the working sequence, but it replaces the incumbent only when its latest completion timestamp is strictly earlier than the fixed completion threshold initialized from the starting solution. A worse candidate is assessed through $\exp[-(f_c - f_{inc})/T]$, where f_c is the candidate objective and f_{inc} is the retained-incumbent objective. Rejection resets the working sequence to the incumbent. The label incumbent-anchored reflects the use of the retained incumbent in the acceptance test and the fixed latest-completion threshold; the retained incumbent is therefore not necessarily the objective-only best solution encountered.

The search starts from the constructive sequence and uses the implemented settings $T_0 = 20$, cooling factor 0.8, five candidate moves per temperature level, and stopping threshold 0.5. The resulting temperature schedule has 17 levels and at most 85 candidate evaluations. Because the two swap positions are sampled independently, a null move can occur when both draws select the same position. The experiment retained one execution per instance and did not store random seeds, candidate-level objectives, infeasibility counts, or convergence traces. The parameters are therefore reported as implemented settings rather than calibrated optima. An infeasible construction cannot replace the feasible incumbent; under the implemented branch logic, it enters the same reset test and was not logged separately.

Algorithm 2 presents the working sequence, retained incumbent, fixed completion threshold, acceptance test, and temperature update in a consistent pseudocode format.

Algorithm 2. Incumbent-anchored simulated annealing used in the study.

Input: initial sequence, deterministic decoder, $T_0 = 20$, $\alpha = 0.8$, $K = 5$, and $T_{min} = 0.5$.
1. Decode the initial sequence; set working = incumbent = initial and record L_0 as the initial latest-completion threshold.
2. Set $T = T_0$.
3. While $T \geq T_{min}$:
4. Repeat K times:
5. Swap two independently sampled positions in working and decode the complete candidate order.
6. If the candidate is feasible and the incumbent is infeasible, set working = incumbent = candidate.
7. Else if the candidate is feasible and $f(\text{candidate}) \leq f(\text{incumbent})$, keep candidate as working and update the incumbent only when $L(\text{candidate}) < L_0$.
8. Else compute $p = \exp[-(f(\text{candidate}) - f(\text{incumbent}))/T]$; reset working to incumbent when $p < u$, otherwise retain the candidate order.
9. An infeasible construction cannot replace the feasible incumbent; under the implemented branch logic, it enters the same reset test and was not counted separately.
10. Set $T = \alpha \times T$.
11. Return the feasible incumbent.

4.4. Lower Bound Estimation

For instances beyond the tractable range of exact optimization, solution quality is assessed against the lower bound in Equations (31) and (32). For each job, Equation (31) gives the minimum number of occupied planning periods implied by its sequential processing workload after machine competition, secondary-resource contention, and inter-stage waiting are relaxed. Equation (32) sums these job-level minima.

$$LB = \sum_{i=1}^{N} MinStay_i \tag{32}$$

Proposition 2. *Validity of the lower bound. The value defined by Equation (32) is a lower bound on the optimal total occupied-period flow time.*

Proof. For each job i, Equation (31) is the ceiling of its sequential processing workload divided by the number of sub-periods per planning period. It omits additional occupied periods caused by machine competition, contention for the continuously retained secondary resource, and inter-stage waiting. Relaxing those sources of delay cannot increase a minimization objective. Therefore, $MinStay_i$ does not exceed the occupied-period span of job i in any feasible schedule, and summing over all jobs yields LB no greater than the optimal total occupied-period flow time. □

The lower bound is evaluated first on the exact-validation configurations by comparison with the proven MILP optimum. For a minimization problem, LB cannot exceed the optimum, and the optimum cannot exceed the objective of a feasible heuristic or SA schedule. On larger instances, the reported SA-to-LB percentage is therefore described only as distance from an optimistic reference, not as an optimality gap.

5. Experiments, Results and Discussion

5.1. Experimental Setup

Instances were generated to span feasible but nontrivial workloads. A horizon that is too short can make an instance infeasible, whereas excessive slack yields trivial schedules.

For stage r, effective capacity is defined in Equation (33) from the planning horizon, sub-period resolution, and the smaller of machine and secondary-resource capacity.

$$\text{Effective Capacity}_r = T\,C\min(E_r, L) \tag{33}$$

Two evaluation scopes are used. The exact-validation scope contains eight configurations for which the MILP supplies proven optima. The larger scope contains 24 configurations beyond this systematic exact benchmark and is used to assess solution quality relative to the analytical lower bound and constructive heuristic.

The study considers 32 configurations defined by the planning horizon, number of jobs, number of stages, machines per stage, and secondary resources. The retained evidence is reported at the configuration-mean level for eight exact-validation configurations and 24 larger configurations. Configurations 1 and 2 use the same processing-time matrices under different sub-period resolutions. The Supplementary File S1 provides the final 32-configuration occupied-period results, exact-reference information, descriptive statistics, and search-parameter documentation.

Processing times were generated as integer sub-period durations from 2 to 20, equivalent to 30 min to 5 h when one sub-period represents 15 min. The C++ implementation used in the study tests each stage workload against 80% of effective stage capacity. Configuration 1 used 24 sub-periods per planning period, whereas Configurations 2–32 used 32. The active resolution was set in the source code for each run; a data-driven input structure is recommended for future implementations. The computational study applies the same-period operation-completion restriction in Constraint (24).

The MILP was solved with IBM ILOG CPLEX 12.5.1 using a two-hour time limit per instance on an Intel i7-7700HQ computer with 16 GB RAM and an NVMe SSD. The heuristic and SA implementation is written in C++. The Supplementary File S2 contains the C++ source supplied with the study materials, the objective-reconciliation details, the prospective logging specification, and the objective-corrected OPL model. The accompanying Excel workbook contains six reader-facing sheets covering study parameters, configuration-level results, the exact-validation configuration scope, exact-solver reference results, descriptive statistics, and the SA search progression.

Formal computation times were recorded for CPLEX, but not separately for the decoder or SA. During the experiments, the constructive heuristic and SA returned results within a few seconds and consistently in less than 10 s on the reported computer. Their runtimes were therefore operationally negligible relative to the CPLEX times, which reached 1740 s, and separate timing instrumentation was not prioritized in the implementation. Because the observation was not captured automatically, it is reported as qualitative context rather than as a formal runtime benchmark.

The implementation was configured to process batches of ten test cases. Because seeds, run-level results, and repeated-run trajectories were not retained, the configuration-level aggregates are the available statistical unit. They describe variation across configurations rather than stochastic variation across repeated runs. Section 6 outlines a replicated, fully logged benchmark for evaluating stochastic variability, timing, and comparative performance.

The evaluation proceeds in three stages. First, the eight exact-validation configuration means are compared with proven MILP optima. Second, the same scope is used to assess SA quality and lower-bound tightness. Third, the 24 larger configuration means are evaluated relative to the analytical lower bound and constructive heuristic.

5.2. Exact-Solver Computation Times

The systematic exact benchmark comprises eight configurations. Table 3 reports the retained mean solution time for each configuration. The first six configurations are solved

within 18 s on average, whereas Configurations 7 and 8 require 279 s and 1740 s. Exact solvability becomes increasingly instance-dependent beyond this systematic benchmark: the Supplementary File S1 also records a proven-optimal Configuration-9 run completed in 1236.45 s. The MILP therefore provides the systematic exact reference for Configurations 1–8 and complementary exact evidence beyond that range.

Table 3. MILP computation times for the tractable configurations.

Instance Size Number	Instance Size	Retained Mean Computing Time (s)
1	*p*2 *sp*24 *j*3 *st*2 *m*2 *r*2	1.19
2	*p*2 *sp*32 *j*3 *st*2 *m*2 *r*2	1.39
3	*p*2 *sp*32 *j*3 *st*3 *m*2 *r*2	1.55
4	*p*2 *sp*32 *j*4 *st*2 *m*2 *r*2	2.24
5	*p*2 *sp*32 *j*5 *st*2 *m*2 *r*2	8.09
6	*p*2 *sp*32 *j*6 *st*2 *m*3 *r*3	17.49
7	*p*3 *sp*32 *j*6 *st*3 *m*3 *r*3	279
8	*p*3 *sp*32 *j*8 *st*3 *m*4 *r*4	1740

The planning horizon is fixed in the computational study. The 80% effective-capacity generation rule limits stage workload to a controlled upper range, while the scheduling procedures themselves retain explicit infeasibility checks. A rolling or variable horizon is a natural extension for dynamic operational planning.

To assess the tractable range, eight configurations vary the planning horizon, number of sub-periods, jobs, stages, machines per stage, and secondary resources. The notation p2 sp24 j3 st2 m2 r2, for example, denotes two planning periods with 24 sub-periods per period, three jobs, two stages, two machines per stage, and two secondary resources. The retained exact-reference evidence is reported as one mean per configuration.

Figure 2 shows the rapid increase in MILP runtime across the exact-validation configurations. Configurations 1–3 are solved in 1.19–1.55 s on average, Configuration 6 requires 17.49 s, and Configurations 7 and 8 require 279 s and 1740 s, respectively. Several dimensions change simultaneously, so this pattern is an aggregate indication of computational burden rather than a controlled scalability experiment for any single factor.

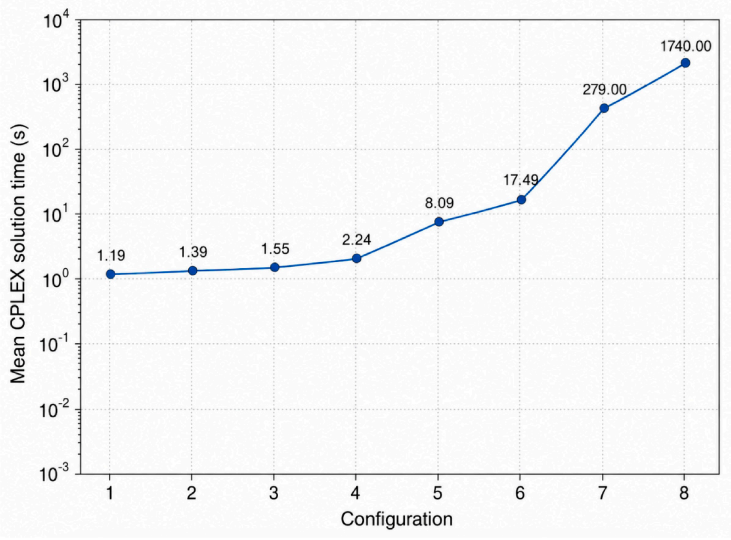

Figure 2. Mean CPLEX solution time by exact-validation configuration, shown on a logarithmic scale.

5.3. Solution Quality Under the Occupied-Period Measure

The constructive heuristic, SA, and lower bound were first evaluated across the eight exact-validation configurations for which the MILP obtained proven optima. Across the eight configuration means, SA is 0.41% above the optimum on average; seven of the eight means equal the optimum. The lower bound is 0.34% below the optimum on average and equals the optimum in six of the eight configuration means. SA improves the constructive heuristic by 0.75% on average in this scope.

Results are reported at the configuration-mean level because instance-level and repeated-run distributions were not retained. Section 5 therefore summarizes the configuration means using medians, sample standard deviations, and ranges. The occupied-period values follow directly from $F = Q + N$, with the fixed job count added consistently to the corresponding transition-count means. The Supplementary File S2 details this relationship, and the workbook provides the configuration values and descriptive statistics.

Figure 3 compares the constructive heuristic, SA, lower bound, and proven optimum across the eight exact-validation configurations under the occupied-period convention. Because the occupied-period measure adds the same fixed job count to all methods within a configuration, the transformation preserves schedule rankings.

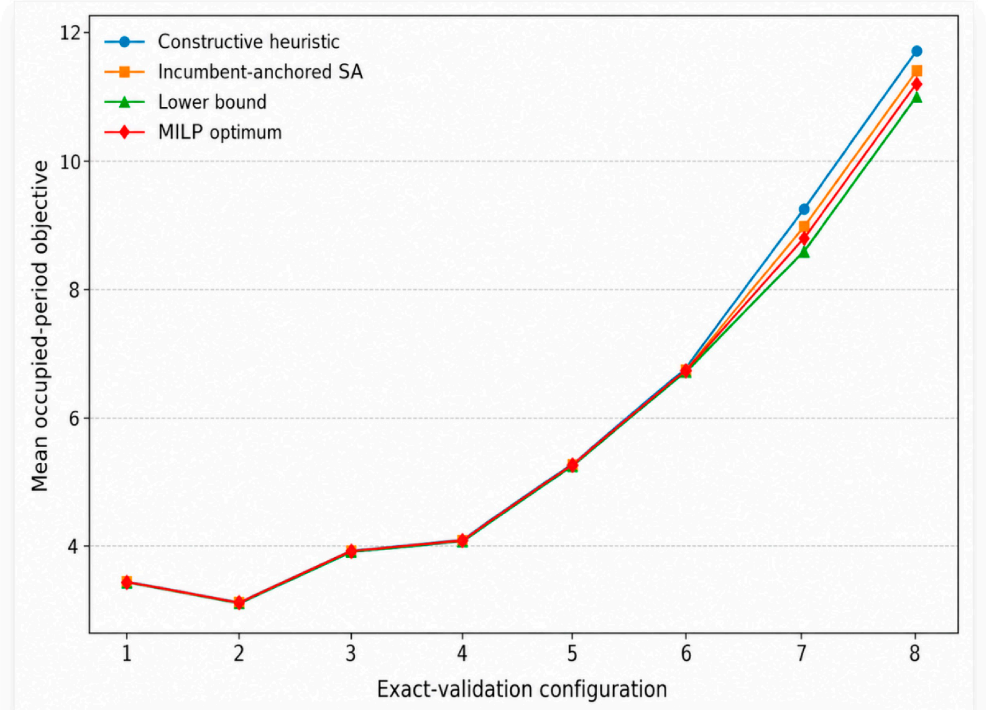

Figure 3. Mean occupied-period objectives for the eight exact-validation configurations.

For Configurations 9–32, the larger-instance analysis uses the lower bound from Section 4.4 as an optimistic reference. Its mean configuration-level deviation below the proven optimum on the exact-validation scope is 0.34%. The Supplementary File S1 also reports a proven-optimal Configuration-9 example with an occupied-period objective of 14.

Across the 24 larger configurations, SA improves the constructive heuristic in 23 configurations and ties it in one; no configuration mean deteriorates. The mean improvement is 3.79%, the median is 3.82%, the sample standard deviation is 2.01 percentage points, and the range is 0.00–8.30% (Table 4). These values describe variation across configurations, not stochastic variation across repeated executions.

The mean SA distance from the analytical lower bound is 10.46% across the larger configurations, with a median of 9.82%, a sample standard deviation of 4.37 percentage points, and a range of 0.00–17.60%. This percentage is not an optimality gap because the optimum is unknown. Because these statistics summarize configuration means rather than repeated independent executions, no confidence interval or hypothesis test is reported.

Table 4. Descriptive computational performance at the configuration-mean level.

Scope	Config. Range	Configurations	Reference	SA Distance: Mean; Median; SD; Range (%)	SA Improvement: Mean; Median; SD; Range (%)
Exact validation	1–8	8	MILP optimum	0.41; 0.00; 1.17; 0.00–3.30	0.75; 0.00; 1.07; 0.00–2.54
Larger	9–32	24	Analytical lower bound	10.46; 9.82; 4.37; 0.00–17.60	3.79; 3.82; 2.01; 0.00–8.30

5.4. Discussion

The exact and larger-instance analyses serve different purposes. On the exact-validation scope, the MILP provides proven optima for assessing both SA solution quality and lower-bound tightness. SA is 0.41% above the optimum on average at the configuration-mean level, while the lower bound is 0.34% below it.

On the larger configurations, the incumbent-anchored SA variant improves 23 of the 24 configuration means and ties one, with a mean improvement of 3.79%. Because candidate schedules are generated by the same deterministic decoder, this comparison quantifies improvement over the constructive sequence. Comparisons with other search engines using the same decoder are proposed in Section 6.

The mean distance to the lower bound is 10.46% on the larger configurations. On the exact-validation scope, the bound averages only 0.34% below the proven optimum and equals it in six of eight configuration means, providing direct evidence of its tightness in the tractable range. As machine competition, retained-resource contention, and waiting increase, the bound becomes more optimistic. The larger-instance distance is therefore interpreted as separation from a relaxed reference rather than an unknown optimality gap.

Grouping the larger configurations by stage count provides additional context. The mean SA-to-bound distance is 5.54% for the three-stage group, 10.01% for the four-stage group, and 14.05% for the five-stage group; the corresponding mean improvements over the constructive heuristic are 3.42%, 3.70%, and 4.21% (Table 5). Because jobs, machines, resources, and horizon also vary, these are descriptive associations rather than causal sensitivity estimates.

Table 5. Descriptive grouping of larger configurations by stage count.

Stages	Configurations	Mean SA-to-Bound Distance (%)	Mean SA Improvement (%)
3	3	5.54	3.42
4	15	10.01	3.70
5	6	14.05	4.21

Complementary groupings from the same 24 larger-configuration means are reported in Supplementary Table S5 and Excel sheet S5. By job-count band, mean SA-to-bound distance was 6.14% for configurations with 10 or fewer jobs, 11.43% for 11–20 jobs, and 12.51% for more than 20 jobs; the corresponding mean improvements over the constructive

heuristic were 2.55%, 4.54%, and 1.87%. By planning horizon, the mean distances were 8.69% for 3–4 periods, 11.02% for 5–7 periods, and 14.83% for horizons longer than seven periods. Grouping by machine and secondary-resource scale produced the same three band counts in this design: mean distance increased from 8.34% in the 4–5 capacity band to 13.35% in the 8-or-more band, while mean improvement ranged from 4.32% to 3.25%. Because the machine and resource bands have identical membership, these two summaries are numerically identical and cannot be interpreted as separable effects. All reported patterns are descriptive associations because several dimensions vary simultaneously; they are not causal sensitivity estimates.

Continuous resource retention changes the operational meaning of waiting. A job that is idle between stages continues to consume secondary-resource capacity, so the occupied-period objective measures a direct consequence of the defining constraint: the planning blocks during which that capacity cannot serve another job. Machine utilization and inter-stage waiting are relevant complementary measures, but the dataset does not include them as separate outcome series.

Primary machines and retained secondary resources must be scheduled jointly because either can become binding over a job's residence time. The formulation and decoder enforce the same feasibility logic during processing and waiting, yielding schedules that account for both types of capacity.

For the larger configurations, the analytical lower bound and constructive heuristic provide formulation-consistent reference points. The Supplementary Files S1 and S2 provide the final configuration-level results, exact-solver reference results, objective reconciliation, descriptive statistics, SA search progression, configuration definitions, and embedded implementation details.

Accordingly, the conclusions concern the continuous-retention formulation, the feasibility of schedules returned by the decoder, the fixed-constant objective reconciliation, exact validation within the tractable scope, and configuration-level solution quality. Run-to-run robustness, convergence behavior, comparative search performance, and instrumented heuristic timing remain subjects for a fully logged benchmark.

6. Conclusions and Future Work

This study considered a hybrid flow-shop problem in which each job passes through multiple stages of identical parallel machines while retaining one renewable secondary resource from first-stage start to final completion, including inter-stage waiting. The objective minimizes total occupied-period flow time. This period-based convention aligns the model with its planning resolution and ensures that a same-period job contributes one occupied period.

The solution framework combines an MILP model, a constructive heuristic with a deterministic resource-aware decoder, and an incumbent-anchored SA variant over job sequences. Across the eight exact-validation configuration means, SA is 0.41% above the proven optimum and the lower bound is 0.34% below it on average. Across the 24 larger configurations, SA improves the constructive heuristic by 3.79% on average, with improvement in 23 configurations and a tie in one.

The paper contributes an occupied-period HFS formulation with continuous retention of one renewable secondary resource during processing and inter-stage waiting, together with a deterministic feasibility-preserving decoder that constructs machine assignments, operation timing, and uninterrupted resource reservations from each job sequence. The incumbent-anchored SA variant searches the resulting sequence space; comparative evaluation against alternative metaheuristics is included in the research agenda below.

The next stage of research will strengthen stochastic validation through stored seeds, repeated runs, run-level statistics, and empirical convergence curves. Comparative testing will retain the same decoder while evaluating the incumbent-anchored rule against conventional current-state simulated annealing and simple baselines under equal evaluation-count and CPU-time budgets. Further decoder development can examine randomized tie breaking and problem-specific insertion, relocation, and critical-resource moves that target earlier release of the continuously retained resource. Stronger congestion-aware lower bounds and multi-objective formulations can then incorporate inter-stage waiting, machine utilization, service criteria, uncertainty, and rolling horizons.

These directions would complement the present contribution, which establishes the continuous-retention formulation, decoder feasibility, exact validation within the tractable scope, and descriptive performance across 32 configurations.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/informatics13120205/s1, File S1: Objective reconciliation, implementation details, computational evidence, and embedded source/model documentation; File S2: Data.

Author Contributions: Conceptualization, N.E.H.S., Z.B., M.A.A. and K.A.-A.; methodology, N.E.H.S., Z.B., M.A.A. and K.A.-A.; formal analysis, N.E.H.S., Z.B., M.A.A. and K.A.-A.; investigation, N.E.H.S., Z.B., M.A.A. and K.A.-A.; resources, N.E.H.S., Z.B., M.A.A. and K.A.-A.; data curation, N.E.H.S., Z.B., M.A.A. and K.A.-A.; writing—original draft preparation, N.E.H.S., Z.B., M.A.A. and K.A.-A.; writing—review and editing, N.E.H.S., Z.B., M.A.A. and K.A.-A.; visualization, N.E.H.S., Z.B., M.A.A. and K.A.-A.; supervision, Z.B.; project administration, N.E.H.S., Z.B., M.A.A. and K.A.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data and supporting methodological materials are provided in the accompanying Supplementary Files S1 and S2. Together, they contain the study parameters, final configuration-level results, exact-validation configuration definitions, exact-solver reference results, descriptive groupings across the instance dimensions, SA search progression, the C++ source supplied with the study materials, and the objective-corrected OPL model.

Acknowledgments: The authors acknowledge the support of the American University of Sharjah under the Open Access Program. The views expressed in this article are those of the authors and do not necessarily represent the position of the American University of Sharjah.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wallrath, R.; Franke, M.; Walter, M. Improved linear programming relaxations for flow shop problems with makespan minimization. *Comput. Oper. Res.* **2025**, *177*, 106970. [CrossRef]
2. Işık, E.E.; Topaloglu Yildiz, S.; Şatır Akpunar, Ö. Constraint programming models for the hybrid flow shop scheduling problem and its extensions. *Soft Comput.* **2023**, *27*, 18623–18650. [CrossRef]
3. Osman, M.A.H.; Ab Rashid, M.F.F.; Nik Mohamed, N.M.Z.; Mu'tasim, M.A.N. Energy-aware scheduling optimization in hybrid flow shops using artificial bee colony algorithm. *J. Mech. Eng. Sci.* **2024**, *18*, 10171–10180. [CrossRef]
4. Çolak, M.; Keskin, G.A. An extensive and systematic literature review for hybrid flowshop scheduling problems. *Int. J. Ind. Eng. Comput.* **2022**, *13*, 185–222. [CrossRef]
5. Mousavi, S.M.; Shahnazari-Shahrezaei, P. Minimizing the makespan and total tardiness in hybrid flow shop scheduling with sequence-dependent setup times. *Manag. Prod. Eng. Rev.* **2023**, *14*, 13–24. [CrossRef]
6. Muñoz-Sánchez, R.; Martínez-Salazar, I.; González-Velarde, J.L.; Ríos Solís, Y.Á. Two hybrid flow shop scheduling lines with assembly stage and compatibility constraints. *PLoS ONE* **2024**, *19*, e0304119. [CrossRef] [PubMed]
7. Liu, Y.; Shen, W.; Zhang, C.; Sun, X. Agent-based simulation and optimization of hybrid flow shop considering multi-skilled workers and fatigue factors. *Robot. Comput.-Integr. Manuf.* **2023**, *80*, 102478. [CrossRef]
8. Costa, A.; Fernandez-Viagas, V.; Framinan, J.M. Solving the hybrid flow shop scheduling problem with limited human resource constraint. *Comput. Ind. Eng.* **2020**, *146*, 106545. [CrossRef]

9. Fleszar, K.; Hindi, K.S. Algorithms for the unrelated parallel machine scheduling problem with a resource constraint. *Eur. J. Oper. Res.* **2018**, *271*, 839–848. [CrossRef]
10. Namakshenas, M.; Braaksma, A.; Mahdavi Mazdeh, M. Minimising total earliness and tardiness with periodically supplied non-renewable resource profiles. *Int. J. Prod. Res.* **2022**, *60*, 3170–3181. [CrossRef]
11. Tran, Q.N.H.; Nguyen, N.Q.; Yalaoui, F.; Amodeo, L.; Chehade, H. Improved formulations and new valid inequalities for a hybrid flow shop problem with time-varying resources and chaining time-lag. *Comput. Oper. Res.* **2023**, *149*, 106018. [CrossRef]
12. Figielska, E. A heuristic for scheduling in a two-stage hybrid flowshop with renewable resources shared among the stages. *Eur. J. Oper. Res.* **2014**, *236*, 433–444. [CrossRef]
13. Mellado-Silva, R.; Cubillos, C.; Cabrera-Paniagua, D.; Urra, E. Flow-shop scheduling problem applied to the planning of repair and maintenance of electromedical equipment in the hospital industry. *Processes* **2022**, *10*, 2679. [CrossRef]
14. AlSerkal, Y.M.; Ibrahim, N.M.; Alsereidi, A.S.; Ibrahim, M.; Kurakula, S.; Naqvi, S.A.; Khan, Y.; Oottumadathil, N.P. Real-time analytics and AI for managing no-show appointments in primary health care in the United Arab Emirates: Before-and-after study. *JMIR Form. Res.* **2025**, *9*, e64936. [CrossRef] [PubMed]
15. van den Broek d'Obrenan, A.; Ridder, A.; Roubos, D.; Stougie, L. Minimizing bed occupancy variance by scheduling patients under uncertainty. *Eur. J. Oper. Res.* **2020**, *286*, 336–349. [CrossRef]
16. Zeiner, H.; Unterberger, R.; Tschuden, J.; Quadri, M.Y. Time-aware optimisation models for hospital logistics. In *Decision Support Systems XIII: Decision Support Systems in an Uncertain World: The Contribution of Digital Twins*; Liu, S., Zaraté, P., Kamissoko, D., Linden, I., Papathanasiou, J., Eds.; Springer: Cham, Switzerland, 2023; pp. 45–55. [CrossRef]
17. Ribas, I.; Leisten, R.; Framiñan, J.M. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Comput. Oper. Res.* **2010**, *37*, 1439–1454. [CrossRef]
18. Neufeld, J.S.; Schulz, S.; Buscher, U. A systematic review of multi-objective hybrid flow shop scheduling. *Eur. J. Oper. Res.* **2023**, *309*, 1–23. [CrossRef]
19. Utama, D.M.; Primayesti, M.D.; Umamy, S.Z.; Kholifa, B.M.N.; Yasa, A.D. A systematic literature review on energy-efficient hybrid flow shop scheduling. *Cogent Eng.* **2023**, *10*, 2206074. [CrossRef]
20. Wang, C.; Pan, Q.-K.; Sang, H.-Y.; Jing, X.-L. A cascaded flowshop joint scheduling problem with makespan minimization: A mathematical model and shifting iterated greedy algorithm. *Swarm Evol. Comput.* **2024**, *86*, 101489. [CrossRef]
21. Rahmativala, S.; Ghahremani, J. A robust optimization method for hybrid flow shop scheduling with uncertain setup times. *Decis. Anal. J.* **2025**, *16*, 100609. [CrossRef]
22. Fu, Y.; Wang, H.; Wang, J.; Pu, X. Multiobjective modeling and optimization for scheduling a stochastic hybrid flow shop with maximizing processing quality and minimizing total tardiness. *IEEE Syst. J.* **2021**, *15*, 4696–4707. [CrossRef]
23. Öztop, H.; Tasgetiren, M.F.; Eliiyi, D.T.; Pan, Q.-K. Iterated greedy algorithms for the hybrid flowshop scheduling with total flow time minimization. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2018), Kyoto, Japan, 15–19 July 2018; pp. 379–385. [CrossRef]
24. Jerbi, B.; Kanoun, S.; Kamoun, H. Multi-objective mathematical programs to minimize the makespan, the patients' flow time, and doctors' workloads variation using dispatching rules and genetic algorithm. *Yugosl. J. Oper. Res.* **2025**, *35*, 365–382. [CrossRef]
25. Zhang, W.; Bao, X.; Hao, X.; Gen, M. Metaheuristics for multi-objective scheduling problems in Industry 4.0 and 5.0: A state-of-the-arts survey. *Front. Ind. Eng.* **2025**, *3*, 1540022. [CrossRef]
26. Fallahpour, Y.; Rafiee, M.; Elomri, A.; Kayvanfar, V.; El Omri, A. A multi-objective planning and scheduling model for elective and emergency cases in the operating room under uncertainty. *Decis. Anal. J.* **2024**, *11*, 100475. [CrossRef]
27. Youn, S.; Geismar, H.N.; Pinedo, M. Planning and scheduling in healthcare for better care coordination: Current understanding, trending topics, and future opportunities. *Prod. Oper. Manag.* **2022**, *31*, 4407–4423. [CrossRef]
28. Figielska, E. Scheduling in a two-stage flowshop with parallel unrelated machines at each stage and shared resources. *Comput. Ind. Eng.* **2018**, *126*, 435–450. [CrossRef]
29. Mlekusch, J.; Hartl, R.F. The dual-resource-constrained re-entrant flexible flow shop: A constraint programming approach and a hybrid genetic algorithm. *Int. J. Prod. Res.* **2025**, *63*, 1803–1824. [CrossRef]
30. Xue, J.; Li, Z.; Zhang, S. Multi-resource constrained elective surgical scheduling with Nash equilibrium toward smart hospitals. *Sci. Rep.* **2025**, *15*, 3946. [CrossRef] [PubMed]
31. Hajji, M.K.; Hamlaoui, O.; Hadda, H. A simulated annealing metaheuristic approach to hybrid flow shop scheduling problem. *Adv. Ind. Manuf. Eng.* **2024**, *9*, 100144. [CrossRef]
32. Tao, X.-R.; Li, J.-Q.; Huang, T.-H.; Duan, P. Discrete imperialist competitive algorithm for the resource-constrained hybrid flowshop problem with energy consumption. *Complex Intell. Syst.* **2021**, *7*, 311–326. [CrossRef]
33. Tao, X.-R.; Pan, Q.-K.; Gao, L. An efficient self-adaptive artificial bee colony algorithm for the distributed resource-constrained hybrid flowshop problem. *Comput. Ind. Eng.* **2022**, *169*, 108200. [CrossRef]
34. Shafiee, M.; Amiri-Aref, M.; Klibi, W. The integration of shared renewable resources considering setup times for the parallel machine scheduling problem. *Comput. Ind. Eng.* **2025**, *200*, 110828. [CrossRef]

35. Yepes-Borrero, J.C.; Villa, F.; Perea, F.; Caballero-Villalobos, J.P. GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources. *Expert Syst. Appl.* **2020**, *141*, 112959. [CrossRef]
36. Abbaszadeh, N.; Asadi-Gangraj, E.; Emami, S. Flexible flow shop scheduling problem to minimize makespan with renewable resources. *Sci. Iran.* **2021**, *28*, 1853–1870. [CrossRef]
37. Wang, M.; Liang, K.; Xiao, D.; Hua, S. Data-driven real-time framework for hybrid flow shop scheduling based on dueling double deep Q network. In Proceedings of the 2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), Nanjing, China, 29–31 March 2024; pp. 1029–1033. [CrossRef]
38. Chen, Z.; Laili, Y.; Zhang, L.; Wang, L. A Trans-Ptr-Nets-based transfer optimization method for multiobjective flexible job-shop scheduling in IIoT. *IEEE Internet Things J.* **2024**, *11*, 25382–25393. [CrossRef]
39. Zhou, J.; Gao, L.; Lu, C.; Li, Y. Knowledge transfer enabled diverse task scheduling for individualized requirements in industrial cloud platform. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 9117–9129. [CrossRef]
40. Xie, H.; Ding, D.; Zhao, L.; Kang, K. Transfer learning based multi-objective evolutionary algorithm for dynamic workflow scheduling in the cloud. *IEEE Trans. Cloud Comput.* **2024**, *12*, 1200–1217. [CrossRef]
41. Zhang, F.; Mei, Y.; Nguyen, S.; Tan, K.C.; Zhang, M. Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling. *IEEE Trans. Cybern.* **2022**, *52*, 10515–10528. [CrossRef] [PubMed]
42. Qin, H.; Li, T.; Teng, Y.; Wang, K. Integrated production and distribution scheduling in distributed hybrid flow shops. *Memetic Comput.* **2021**, *13*, 185–202. [CrossRef]
43. Gupta, J.N.D. Two-stage, hybrid flowshop scheduling problem. *J. Oper. Res. Soc.* **1988**, *39*, 359–364. [CrossRef]
44. Hoogeveen, J.A.; Lenstra, J.K.; Veltman, B. Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard. *Eur. J. Oper. Res.* **1996**, *89*, 172–175. [CrossRef]
45. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discret. Math.* **1979**, *5*, 287–326. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.