

Article

The Remote Sensing Data Transmission Problem in Communication Constellations: Shop Scheduling-Based Model and Algorithm

Jiazhao Yin ¹, Yuning Chen ^{1,*}, Xiang Lin ¹ and Qian Zhao ²¹ College of Systems Engineering, National University of Defense Technology, Changsha 410073, China; yinjiazhao@nudt.edu.cn (J.Y.); linxiang18@alumni.nudt.edu.cn (X.L.)² Beijing Institute of Remote Sensing Information, Beijing 100192, China; zhaoqian_nudt@163.com

* Correspondence: yuning_chen@nudt.edu.cn; Tel.: +86-183-9080-3313

Abstract

Advances in satellite miniaturisation have led to a steep rise in the number of Earth-observation platforms, turning the downlink of the resulting high-volume remote-sensing data into a critical bottleneck. Low-Earth-Orbit (LEO) communication constellations offer a high-throughput relay for these data, yet also introduce intricate scheduling requirements. We term the associated task the Remote Sensing Data Transmission in Communication Constellations (DTIC) problem, which comprises two sequential stages: inter-satellite routing, and satellite-to-ground delivery. This problem can be cast as a Hybrid Flow Shop Scheduling Problem (HFSP). Unlike the classical HFSP, every processor (e.g., ground antenna) in DTIC can simultaneously accommodate multiple jobs (data packets), subject to two-dimensional spatial constraints. This gives rise to a new variant that we call the Hybrid Flow Shop Problem with Two-Dimensional Processor Space (HFSP-2D). After an in-depth investigation of the characteristics of this HFSP-2D, we propose a constructive heuristic, denoted NEHedd-2D, and a Two-Stage Memetic Algorithm (TSMA) that integrates an Inter-Processor Job-Swapping (IPJS) operator and an Intra-Processor Job-Swapping (IAJS) operator. Computational experiments indicate that when TSMA is initialized with the solution produced by NEHedd-2D, the algorithm attains the optimal solutions for small-sized instances and consistently outperforms all benchmark algorithms across problems of every size.

Keywords: remote sensing satellite data transmission; hybrid flow shop scheduling; memetic algorithm; satellite constellation



Academic Editor: Sotirios K. Goudos

Received: 24 July 2025

Revised: 21 August 2025

Accepted: 25 August 2025

Published: 1 September 2025

Citation: Yin, J.; Chen, Y.; Lin, X.; Zhao, Q. The Remote Sensing Data Transmission Problem in Communication Constellations: Shop Scheduling-Based Model and Algorithm. *Technologies* **2025**, *13*, 384. <https://doi.org/10.3390/technologies13090384>

Copyright: © 2025 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 (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Remote sensing satellites are indispensable to contemporary Earth-observation systems, supporting disaster mitigation, crop yield estimation, weather forecasting, and many other applications. A typical mission cycle comprises five steps: (i) ground control centres generate observation schedules; (ii) ground antennas upload the schedules to the satellites; (iii) the satellites acquire imagery accordingly; (iv) the acquired data are downlinked; and (v) analysts process the products. Among these steps, data transmission, consisting of the delivery of observation data to the ground through direct space-ground links or relay satellites, remains the key bottleneck for timely data exploitation [1–4]. Satellite miniaturisation and falling launch costs have accelerated the deployment of commercial low-Earth-orbit (LEO) imaging constellations; although these larger fleets greatly enhance observation opportunities, they also generate data at an unprecedented pace; thus, efficient downlinking

has become the limiting factor in the value chain. Existing transmission paradigms are broadly divided into ground station-based and geostationary relay-based modes.

In the ground station mode, illustrated in Figure 1, satellites transmit their data directly to fixed ground stations. Scheduling research on this mode began with the seminal hybrid integer-programming model for the Air Force Satellite Control Network (AFSCN) proposed by Gooley et al. [5]. Since then, a rich body of heuristics and meta-heuristics has emerged: Chen et al. [6] designed a fast local-search heuristic that jointly plans imaging and transmission; Zhang and Xing [7] improved genetic algorithms through refined encoding and decoding; and swarm intelligence-based approaches have also proven effective [8,9]. Direct downlinks feature high data rates, negligible onboard energy constraints, and modest operating costs. Their chief limitation is geometric; a ground station can communicate only while the satellite passes overhead, meaning that the contact windows are too short to clear the massive data backlogs of prominent constellations.

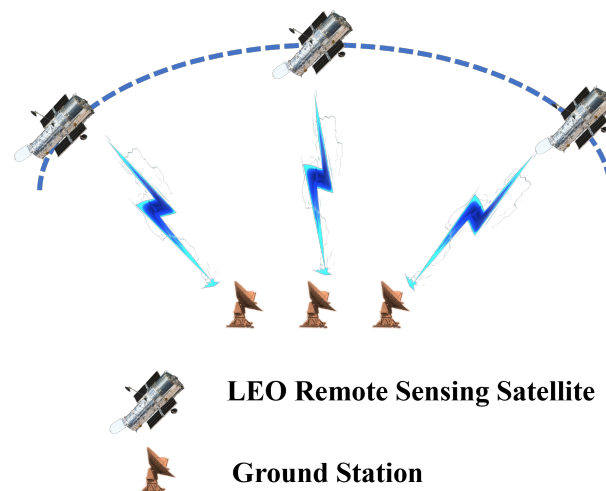


Figure 1. Ground station-based data transmission process.

In the relay satellite mode, shown in Figure 2, an LEO image satellite first uplinks its data to a geostationary (GEO) relay, which then forwards the packets to ground stations. The GEO platform's high altitude affords almost continuous visibility, enlarging the transmission window. Notable studies include the breakpoint-resumption scheme combined with adaptive large neighbourhood search by Chen et al. [10], the conflict-resolution-aided iterative scheduler (CRITS) by Wu et al. [11], and several deep reinforcement learning-based frameworks [12]. However, these approaches are hampered by the high cost of GEO launches, the relay's limited ability to serve multiple imaging satellites concurrently, and its lower effective downlink rate compared with direct ground stations.

The recent rise of large-scale LEO communication constellations such as Starlink and OneWeb offers a promising third paradigm [13,14]. As sketched in Figure 3, remote sensing satellites upload data to an ingress communication satellite inside a precomputed visibility window; the data then traverse the constellation's inter-satellite links until an egress satellite downlinks them to a terrestrial gateway. The sheer number of communication satellites yields abundant contact opportunities, while the comparatively low cost of LEO platforms delivers economic advantages over dense ground station networks or expensive GEO relays. Distinct differences nonetheless arise. For instance, a gateway can receive data from several egress satellites simultaneously under bandwidth constraints, whereas a traditional ground antenna usually tracks only one spacecraft at a time; in addition, unlike the single-layer decision in GEO relay scheduling, the LEO constellation requires two-layer planning to choose both the gateway and the appropriate contact windows. Therefore, conventional scheduling techniques cannot be applied directly, and new models are required.

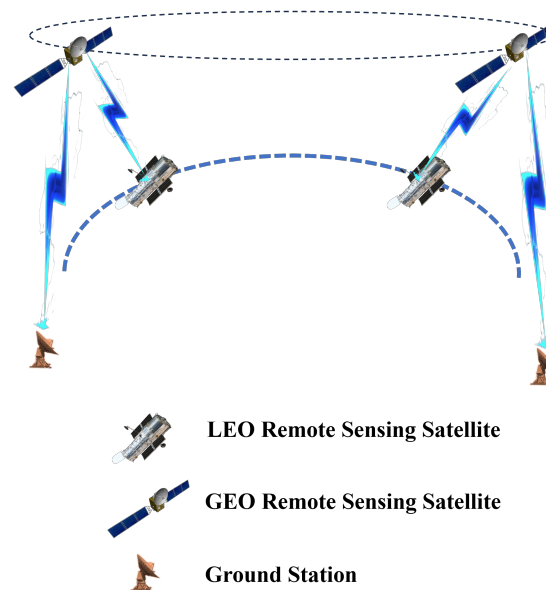


Figure 2. Relay satellite-based data transmission process.

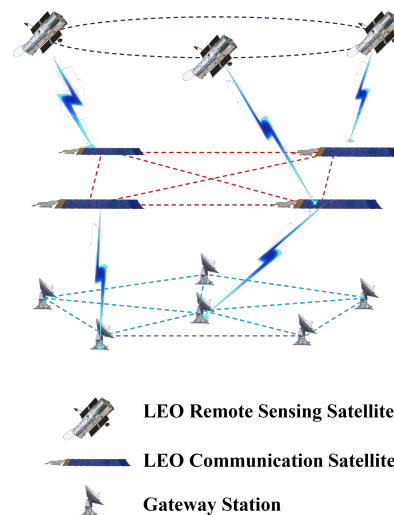


Figure 3. Data transmission in communication constellations.

We decompose the resulting Data Transmission In Communication Constellations (DTIC) problem into an inter-satellite stage followed by a satellite-to-gateway stage. Interpreting data bundles as jobs, communication satellites as first-stage machines, and gateways as second-stage machines, the resulting problem exhibits unidirectional flow and uncertain routing. The DTIC problem introduces two fundamental novelties that are absent in classical scheduling. (1) Integrated two-layer decisions: Unlike traditional satellite scheduling (e.g., AFSCN models) that optimize antenna assignment alone, DTIC requires joint routing selection (inter-satellite) and delivery scheduling (satellite-to-ground) with coupled constraints. (2) Two-dimensional resource contention: Gateway stations exhibit continuous spatial resource sharing (bandwidth allocation) rather than binary machine occupancy, extending beyond classical HFS models.

Regarding the selection of a suitable scheduling model, we note that although Job Shop Scheduling permits flexible routing, it cannot enforce the DTIC problem's stage-precedence requirement, under which inter-satellite transmission must occur before downlink. Open Shop Scheduling is likewise inappropriate, as its allowance for arbitrary operation ordering conflicts with the DTIC problem's fixed sequence. In contrast, the flow-shop structure

of the Hybrid Flow Shop Problem (HFSP) inherently matches our two-stage process, and its hybrid extension with parallel machines at each stage naturally represents the multiple communication satellites in Stage 1 and multiple gateways with spatial capacity in Stage 2. Consequently, the DTIC problem is best modeled within the Hybrid Flow Shop Scheduling (HFSS) framework, which is the canonical paradigm for multi-stage systems featuring identical parallel processors and a unidirectional flow of operations [15]. A crucial departure from the classical HFSP is the many-to-one satellite-to-gateway downlink governed by continuous bandwidth allocations. To capture this feature, we introduce the notion of processor space, which combines temporal and spatial constraints; we formalise the resulting variant as the Hybrid Flow Shop Problem with Two-Dimensional Processor Space (HFSP-2D).

Solution strategies for the HFSP can be divided into exact and heuristic methods. Exact algorithms, such as the branch-and-bound schemes of Choi and Lee [16] and their enhancements by Lee [17], guarantee optimality but scale only to small instances, rendering them impractical for the DTIC problem's massive task sets. Consequently, heuristic research has flourished. The Nawaz–Enscore–Ham (NEH) constructive heuristic [18] remains the cornerstone of many contemporary variants, including extensions for distributed flow shops [19,20], decomposed parallel-machine subproblems [21], and sequence-dependent setups [22,23]. Most existing work, however, targets makespan minimisation, whereas DTIC seeks to minimise total tardiness, an objective that has received far less attention in the HFSP literature [24]. Although makespan minimisation is the prevalent objective in manufacturing, total tardiness constitutes a more pertinent performance measure for remote sensing data transmission. In applications such as maritime search and rescue, emergency response, and early warning of major natural disasters, the usefulness of a remote sensing product is determined by the latency between its acquisition and its delivery to the ground segment; data received after the prescribed deadline loses its operational value. Accordingly, this study takes the minimisation of the to transmission tardiness as its optimisation objective. In this context, effective tie-breaking among equal-cost schedules is essential, yet remains largely unexplored.

This study makes three principal contributions. First, at the problem level, we formalise the HFSP-2D and present a Mixed-Integer Linear Programming (MILP) model that embeds all key inter-satellite and satellite-to-gateway constraints while permitting multiple processors per stage. Small instances solved with CPLEX confirm the model's correctness. Second, at the algorithmic level, we design an NEHedd-2D constructive heuristic with domain-specific tie-breakers and develop a Two-Stage Memetic Algorithm (TSMA) meta-heuristic that incorporates two specially designed local-search operators. Third, at the empirical level, comprehensive experiments assess each algorithmic component and benchmark TSMA against state-of-the-art alternatives, demonstrating its superior efficiency and solution quality.

The rest of this paper is organized as follows: Section 2 contains the problem description; the proposed NEHedd-2D and TSMA are introduced in Section 3; and our experiments are presented in Section 4.

2. Model

2.1. Nomenclature

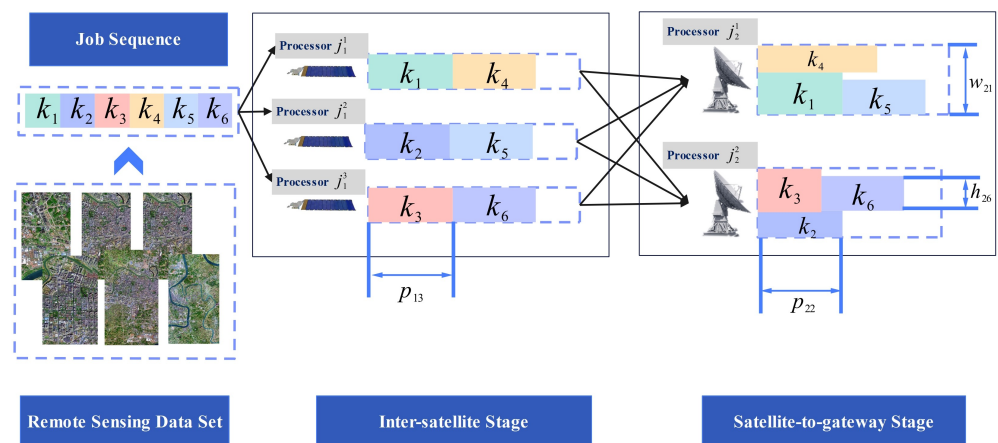
The parameters in the model are described in Table 1.

Table 1. Nomenclature.

Symbols	Definition
i	processing stage, $i \in I = \{1, \dots, m\}$
j	processor in stage i , $j \in J_i = \{1, \dots, n_i\}$
k	job, $k \in K = \{1, \dots, v\}$
π	job list, $\pi = \{k_1, k_2, \dots, k_v\}$
π_{ij}	job list of processor j in stage i
m	number of processing stage
v	number of jobs
n_i	number of parallel processors in stage i
p_{ik}	processing time for job k in stage i
r_{ik}	spacing requirements of job k in stage i
Q	a large number not less than the maximum completion Time
P	a large number not less than the maximum space of processors
M	a large number not less than 4
T	total tardiness
c_{ik}	completion time of job k in stage i
d_{ik}	due time of job k in stage i
h_{ik}	start of spacing requirements of job k in stage i
w_{ij}	space of processor j in stage i
x_{ijk}	1, if job k is assigned to processor $j \in J_i$ in stage $i \in I$; otherwise $x_{ijk} = 0$
y_{ikl}	1, if job k precedes job l ; otherwise $y_{ikl} = 0$
z_{ikl}	1, if job k is processed below job l ; otherwise $z_{ikl} = 0$
μ_{ikl}	1, if job k and job l in stage $i \in I$ do not conflict under resource constraints; otherwise $\mu_{ikl} = 0$
δ_{ijkl}	1, if job k and job l is assigned to processor $j \in J_i$ in stage $i \in I$; otherwise $\delta_{ijkl} = 0$

2.2. Problem Description

As illustrated in Figure 4, the DTIC problem can be described as follows. The remote sensing data set acquired by satellites is arranged according to predefined rules into a job sequence that contains v jobs, $K = \{k_1, k_2, \dots, k_v\}$. In the inter-satellite stage, the jobs are assigned to n_1 processors, corresponding to the LEO communication satellites, and the processing times of different jobs are not allowed to overlap. In the satellite-to-gateway stage, the jobs are dispatched to n_2 processors, corresponding to the gateway stations, where multiple jobs may be processed simultaneously provided that the processing space constraint (i.e., the bandwidth of the gateway stations) is satisfied.

**Figure 4.** Two-stage transmission architecture of the remote sensing data transmission process.

Consequently, the problem is naturally partitioned into two stages, namely, classical hybrid flow-shop scheduling and hybrid flow-shop scheduling incorporating processing space constraints. It is worth noting that the classical HFSP can be regarded as a special case of the latter in which the processing space requirement of every job is identical and equal to the whole processing space of all processors in the stage. Therefore, the DTIC problem is rigorously formulated as an HFSS problem with processing space constraints; its processing sequence is extended from one to two dimensions, and the problem is henceforth denoted as the Hybrid Flow Shop Problem with Two-Dimensional Processor Space (HFSP-2D). An illustrative Gantt chart of the resulting schedule is provided in Figure 5.

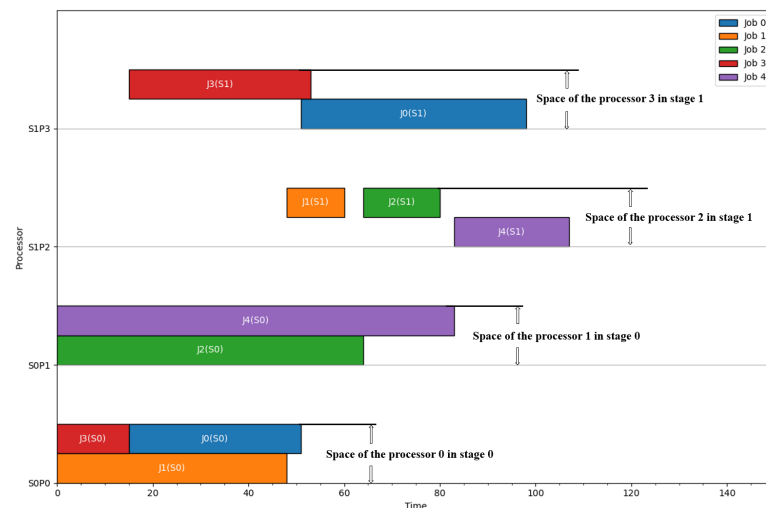


Figure 5. Gantt chart of the resulting schedule for HFSP-2D.

The correspondence relationships between various elements of the DTIC and HFSP-2D problems are presented in Table 2.

Table 2. Correspondence relationships between the DTIC and HFSP-2D problems.

DTIC Elements	HFSP-2D Elements
Remote sensing data transmission job	Job
LEO communication satellite	Processor in first stage
Ground gateway station	Processor in second stage
Maximum bandwidth of gateway station	Processor maximum processing space
Data transmission rate	Job space requirement

2.3. Mixed-Integer Linear Programming Formulation

This subsection presents a Mixed-Integer Linear Programming (MILP) model for the HFSP-2D. The notation used in the sequelae is identical to that introduced in Section 2.1. The model minimises the total tardiness while guaranteeing feasible temporal and spatial arrangements of all jobs.

2.3.1. Objective Function

Minimise the sum of individual job tardiness values:

$$\min \sum_{k=1}^n T_k. \quad (1)$$

2.3.2. Tardiness Definition

$$T_k \geq C_{mk} - d_k \quad \forall k \in K \quad (2)$$

$$T_k \geq 0 \quad \forall k \in K \quad (3)$$

Constraints (2) and (3) state that the tardiness T_k of job k is the positive part of the difference between its completion time at the last stage C_{mk} and its due date d_k .

2.3.3. Job–Processor Assignment

Each job is processed by exactly one processor at every stage:

$$\sum_{j \in J_i} x_{ijk} = 1, \quad \forall i \in I, \forall k \in K, |J_i| > 1. \quad (4)$$

2.3.4. Stage-Precedence (Completion) Constraints

Processing at stage i can only start after the preceding stage has finished.

$$c_{1k} \geq p_{1k} \quad \forall k \in K \quad (5)$$

$$c_{ik} - c_{(i-1)k} \geq p_{ik} \quad \forall i \in I, i > 1, \forall k \in K \quad (6)$$

2.3.5. Processor-Capacity (Spatial) Constraints

The right-most occupied position of a job may not exceed the available width w_{ij} of the assigned processor j :

$$(h_{ik} + r_{ik}) - P(1 - x_{ijk}) \leq w_{ij}, \quad \forall i \in I, \forall j \in J_i, \forall k \in K. \quad (7)$$

2.3.6. Spatiotemporal Non-Overlap Constraints

Jobs assigned to the same processor must overlap in neither time nor space. The following pairs of constraints enforce temporal ((8) and (9)) and spatial ((10) and (11)) separation, respectively.

$$c_{ik} + Q(2 + y_{ikl} - \delta_{ijkl} - \mu_{ikl}) \geq c_{il} + p_{ik}, \quad \forall i \in I, \forall j \in J_i, \forall k < l \in K \quad (8)$$

$$c_{il} + Q(3 - y_{ikl} - \delta_{ijkl} - \mu_{ikl}) \geq c_{ik} + p_{il}, \quad \forall i \in I, \forall j \in J_i, \forall k < l \in K \quad (9)$$

$$h_{ik} + P(1 + z_{ikl} - \delta_{ijkl} + \mu_{ikl}) \geq h_{il} + r_{ik}, \quad \forall i \in I, \forall j \in J_i, \forall k < l \in K, w_{ij} > 0 \quad (10)$$

$$h_{il} + P(2 - z_{ikl} - \delta_{ijkl} + \mu_{ikl}) \geq h_{ik} + r_{il}, \quad \forall i \in I, \forall j \in J_i, \forall k < l \in K, w_{ij} > 0 \quad (11)$$

2.3.7. Linking Constraint

Constraint (12) activates the foregoing spati-temporal constraints only when the two jobs under consideration are processed on the same processor:

$$x_{ijk} + x_{ijl} \leq 1 + M\delta_{ijkl}, \quad \forall i \in I, \forall j \in J_i, \forall k < l \in K, w_{ij} > 0. \quad (12)$$

Collectively, constraints (8)–(12) guarantee that any two jobs sharing a processor are sequenced without temporal interference and are placed within the processor's horizontal workspace without spatial collision.

2.4. Complexity Analysis of the MILP Model

The classical hybrid flow-shop scheduling problem is already NP-hard even in the restricted case of two stages with a single processor at each stage [25]; therefore, the HFSP-

2D inherits this hardness. In the worst case, solving the proposed MILP to optimality is exponential in the number of binary variables $|\mathcal{B}|$.

Let $n_{\max} = \max n_i$ be the maximum number of jobs at any stage. The model contains n continuous tardiness variables T_k along with $2mn$ continuous timing/position variables c_{ik} and h_{ik} , i.e., $\mathcal{O}(mn)$ continuous variables in total. The binary part comprises three groups: (i) assignment variables x_{ijk} , which have cardinality $\sum_i s_i n = \mathcal{O}(ms_{\max}n)$; (ii) pair-dependent variables $y_{ikl}, z_{ikl}, \mu_{ikl}$ introduced for every ordered job pair at each stage, providing $3m\binom{n}{2} = \mathcal{O}(mn^2)$ variables; and (iii) variables δ_{ijkl} that additionally distinguish processors, amounting to $\sum_i s_i \binom{n}{2} = \mathcal{O}(ms_{\max}n^2)$.

Hence, the total number of binary variables is $|\mathcal{B}| = \mathcal{O}(ms_{\max}n^2)$. Despite this polynomial growth in model size, the NP-hardness of the underlying problem implies that solving the MILP requires exponential time in the worst case; specifically, the search space scales as $(2^{\mathcal{O}(ms_{\max}n^2)})$, which becomes intractable for moderate n .

Because both the number of binary variables and the total number of constraints grow quadratically with n , optimally solving instances with $n \geq 30$ and moderate m is already beyond the reach of state-of-the-art MILP solvers. Therefore, for larger problems we must rely on meta-heuristic strategies.

It should be noted that while the MILP provides an exact formulation for the HFSP-2D, its computational complexity inherently limits its applicability to large-scale scenarios. Alternative exact approaches, such as branch-and-price or logic-based Benders decomposition, could potentially improve scalability; we explicitly recommend this as future work. However, because data transmission scheduling must be carried out in real time and must accommodate large constellations of remote sensing and communication satellites, our primary focus remains on developing fast and effective heuristic algorithms instead of on refining exact methods. In this study, we treat the MILP solution as an upper bound for the problem, and use it to validate the correctness of our heuristic algorithm.

3. Memetic Algorithm for HFSP-2D

Memetic algorithms have proved to be highly efficient not only for shop scheduling problems [26] but also for remote sensing satellite mission planning problems [27,28]. To tackle the HFSP-2D, we propose a Two-Stage Memetic Algorithm (TSMA). As illustrated in Figure 6, the proposed Two-Stage Memetic Algorithm (TSMA) comprises four principal phases: initialization, crossover, mutation, and local search. This iterative cycle of construction, perturbation, and refinement progressively guides the search toward high-quality solutions for the HFSP-2D.

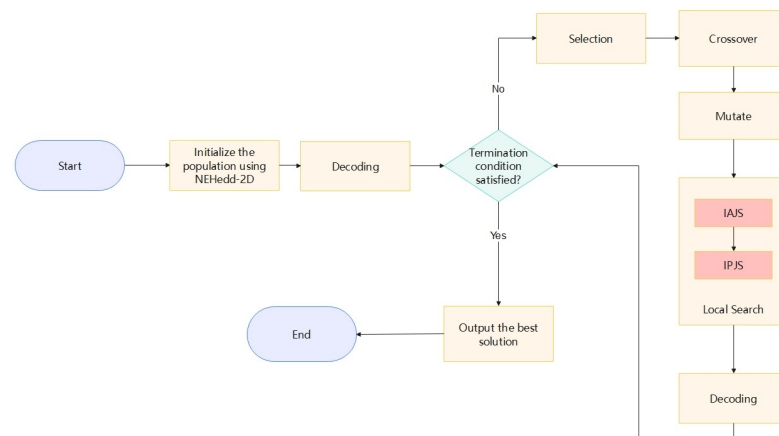


Figure 6. Flowchart of the TSMA.

3.1. Encoding and Decoding

3.1.1. Encoding

During the encoding stage, a single chromosome is produced that concurrently records the global processing order $\pi = \{k_1, k_2, \dots, k_v\}$. For every processor j in stage i , the local subsequence π_{ij} is inherited from π , providing a compact yet comprehensive description that captures both the shop-wide dispatching sequence and the exact execution order on each processor.

3.1.2. Decoding

During decoding, the completion times of all jobs are computed stage-by-stage.

As illustrated in Algorithm 1, in first stage, the overall job sequence π is first used to obtain the first-stage workspace-unconstrained completion time c_{1k} for each job.

Algorithm 1 Decoding for first stage (Decode1)

Require: Global job order $\pi = \langle k_1, k_2, \dots, k_v \rangle$; number of parallel processors n_1 ; processing times $p[\cdot][1]$

Ensure: Completion-time vector $C_1 = \langle c_{11}, c_{12}, \dots, c_{1v} \rangle$

```

1: for  $b \leftarrow 1$  to  $n_1$  do Ready[ $b$ ]  $\leftarrow 0$ 
2: Build a min-heap  $H$  of pairs (Ready[ $b$ ],  $b$ )
3: for  $a \leftarrow 1$  to  $v$  do
4:    $(t, b) \leftarrow \text{ExtractMin}(H)$ 
5:    $f \leftarrow t + p[k_a][1]$ 
6:   Ready[ $b$ ]  $\leftarrow f$ 
7:   Insert( $H$ , (Ready[ $b$ ],  $b$ ))
8:    $c_{1a} \leftarrow f$ 
9: end for
10: return  $C_1$ 

```

As illustrated in Algorithm 2, in the second stage, the per-processor job sequences π_{ij} and first-stage completion times together yield the second-stage completion time c_{2k} ; afterwards, the total tardiness $\sum T$ is calculated from c_{2k} and the due date d_k of each job.

Algorithm 2 Decoding for second stage (Decode2)

Require: Completion times in first stage $C_1 = \langle c_{11}, c_{12}, \dots, c_{1v} \rangle$; processing times $p[\cdot][2]$; due date $D = \langle d_1, d_2, \dots, d_v \rangle$; job lists π_{2j} for every processor j in second stage, $j = 1, \dots, n_2$.

Ensure: Total tardiness T

```

1:  $T \leftarrow 0$ 
2: for  $j \leftarrow 1$  to  $n_2$  do
3:   for all  $k \in \pi_{2j}$  in given order do
4:      $s \leftarrow \max(c_{1k}, \text{BL}(j, k))$ 
5:      $f \leftarrow s + p[k][2]$ 
6:      $T \leftarrow T + \max(f - d_k, 0)$ 
7:   end for
8: end for
9: return  $T$ 

```

Due to the processing space constraints around concurrent job processing, the FAM rule cannot directly determine the scheduling sequence on a single processor. Therefore, this paper introduces the Bottom-Left Placement (BL) algorithm (Algorithm 3) to schedule the processing sequence of jobs on individual processors.

Algorithm 3 Bottom-left placement (BL)

Require: process space of processor w ; spacing requirements of job r ; processing time of job p ; current set of placed jobs $R = \{(x_j, x_j + p_j, y_j, y_j + r_j) \mid j = 1, \dots, m\}$

Ensure: completion time of new job c , update R

- 1: $x \leftarrow \max(x_1, x_2, \dots, x_m)$, $y \leftarrow w - p$
- 2: **while** $(x, x + p, y, y + r) \cap R = \emptyset$ **do**
- 3: $x \leftarrow x - 1$
- 4: $y \leftarrow y - 1$
- 5: **end while**
- 6: Insert vertex $(x, x + p, y, y + r)$ into R in sorted order
- 7: $c \leftarrow x + p$
- 8: **return** c

3.2. Initialization

To expedite the generation of high-quality initial solutions, we extend the classical constructive NEHedd heuristic to the two-dimensional domain and introduce the resulting NEHedd-2D algorithm.

NEHedd, which sequences jobs using the Earliest Due Date (EDD) rule, is a well-known constructive heuristic for minimizing total tardiness in the Hybrid Flow Shop Problem (HFSP) [29]. Nevertheless, its scope is limited to one-dimensional scheduling. In contrast, the HFSP-2D simultaneously requires assigning jobs to parallel machines at every stage and allocating the secondary resource shared by the jobs processed on the same machine. To meet these additional requirements, we replace the profit evaluation step in the original NEHedd with a tailored two-dimensional decoding procedure. The resulting algorithm is designated NEHedd-2D (Algorithm 4). Because NEHedd-2D's iterative insertion phase maintains only a single global processing sequence π , jobs are ultimately dispatched to individual machines according to the First-In, First-Out (FIFO) and First Available Machine (FAM) rules.

Algorithm 4 NEHedd-2D

Require: $\pi \leftarrow$ Jobs ordered by non-decreasing due dates, where $\pi \leftarrow \{k_1, k_2, \dots, k_v\}$;

Ensure: overall job sequence π_v ; job sequence set $\pi_{2,j}$ on each processor $j = 1, \dots, n_2$

- 1: $\pi_1 \leftarrow \{k_1\}$
- 2: **for** $k = 2$ to v **do**
- 3: Test job k in any possible position i of π_k in first stage.
- 4: $C \leftarrow \text{Decode1}(\pi)$
- 5: **for** $j \leftarrow 1$ to n_2 **do**
- 6: Test job k in any possible position i of $\pi_{2,j}$, $j \in J_2$.
- 7: $T \leftarrow \text{Decode2}(\pi)$
- 8: **end for**
- 9: Select the best processor j and best insertion position in processor j .
- 10: Insert k to $\pi_{2,j}$
- 11: Select the best insertion position in π_{k-1} .
- 12: Insert k to π_{k-1} .
- 13: **end for**

return π_v and $\pi_{2,j}$

3.3. Selection

The selection operator in this work is implemented using an elitist tournament scheme.

At the start of each generation, the algorithm first invokes elitism. A small elite subset consisting of the best-scoring individuals is copied unchanged into the next population, thereby safeguarding the highest-quality genetic material.

The remaining positions are filled via tournament selection. For every vacancy, a group of candidates is sampled uniformly at random from the current population. The candidate with the lowest objective value in a minimisation context wins the tournament and is placed in the offspring pool. This procedure is repeated until the population is fully replenished.

By coupling elitism with stochastic tournaments, this strategy simultaneously preserves outstanding solutions. It sustains selection pressure and diversity, thereby promoting both rapid convergence and compelling global exploration within the memetic algorithm.

3.4. Crossover

As illustrated in Figure 7, the crossover operator in our algorithm works in three consecutive phases. It first identifies the crossover positions on the task permutation π , then marks in the other parent the genes (tasks) that will not be exchanged, and finally copies all remaining genes in their original order into the crossover segment of the opposite parent.

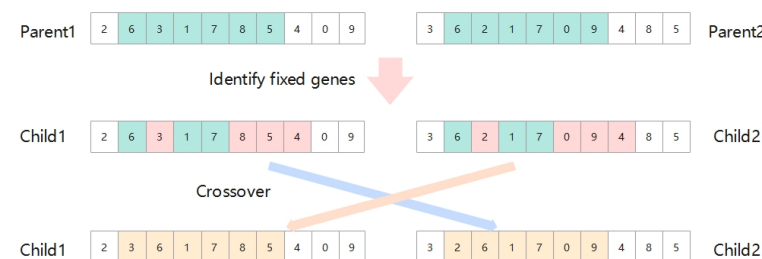


Figure 7. Crossover.

A set of crossover points is uniformly sampled along the chromosome; the number of points is even and lies strictly between the two extremes of single-point and whole-length crossover, while the sampled positions are sorted in ascending order. These points partition each chromosome into segments that are swapped in an alternating “keep and swap” fashion, allowing the offspring to inherit contiguous blocks of genes that preserve positional information while still recombining material from both parents.

Because each permutation must contain every task exactly once, the tasks in the parents that are not exchanged are explicitly identified, preventing duplicate or missing tasks during the crossover process.

3.5. Mutate

The employed mutation operator is a permutation-preserving swap mutation. Two distinct loci are chosen uniformly at random within the chromosome, and the genes located at these loci are exchanged. This simple exchange perturbs the current solution while maintaining the feasibility of the permutation, thereby enhancing population diversity and reducing the risk of premature convergence.

3.6. Local Search

To further optimize the scheduling scheme, we designed two neighborhood search methods for processing space scheduling. The primary approach is to reduce idle time in the production space by relocating jobs between processors or adjusting job sequences within individual processors, thereby minimizing the total tardiness cost.

3.6.1. Inter-Processor Job Swapping Operator (IPJS)

Because the NEHedd-2D algorithm schedules jobs using FIFO and FAM rules, the order in which jobs enter the production flow determines the processor assignment for

each job at every stage. Exchanging jobs between processors can expand the search space. The Inter-Processor Job Swapping (IPJS) procedure is shown in Algorithm 5 and Figure 8.

Algorithm 5 Inter-processor job swapping operation

Require: Set of processors J at second stage; current schedule of processors π_{2j}

Ensure: A new schedule of processors π'_{2j} at the second stage if the total tardiness does not increase

- 1: Randomly pick two distinct machines j_a, j_b
 - 2: Randomly choose start indices s_a, s_b of π_a, π_b
 - 3: Determine crossover lengths $l_a \in (0, \text{size}(\pi_{2a}) - s_a), l_b \in (0, \text{size}(\pi_{2b}) - s_b)$ from s_a, s_b
 - 4: $idx_1 \leftarrow \pi_{2a}[s_a], idx_2 \leftarrow \pi_{2b}[s_b]$
 - 5: $seg_1 \leftarrow \pi_{2a}[idx_1, \dots, idx_1 + l_1 - 1], seg_2 \leftarrow \pi_{2b}[idx_2, \dots, idx_2 + l_2 - 1]$
 - 6: Delete seg_1 from π_{2a} and seg_2 from π_{2b}
 - 7: $\pi'_{2a} \leftarrow \text{Insert } seg_2 \text{ at } idx_1 \text{ in } \pi_{2a}, \pi'_{2ab} \leftarrow \text{Insert } seg_1 \text{ at } idx_2 \text{ in } \pi_{2b}$
 - 8: $T_a \leftarrow \text{decode2}(\pi_{2a}), T_b \leftarrow \text{decode2}(\pi_{2b})$
 - 9: $T'_a \leftarrow \text{decode2}(\pi'_{2a}), T'_b \leftarrow \text{decode2}(\pi'_{2b})$
 - 10: **if** $T'_a + T'_b > T_a + T_b$ **then**
 - 11: $\pi_{2a} \leftarrow \pi'_{2a}$
 - 12: $\pi_{2b} \leftarrow \pi'_{2b}$
 - 13: **end if**
- return** A new schedule of processors π'_{2j}
-

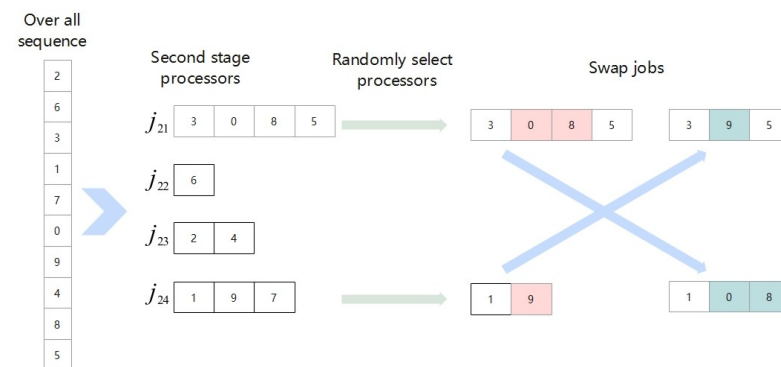


Figure 8. IPJS operation.

3.6.2. Intra-Processor Job Swapping Operator (IAJS)

In the NEHedd-2D algorithm, the job processing sequence on a single processor is determined by job arrival times or job completion times from the previous stage. Different job processing sequences can affect job waiting times. Therefore, we designed an Intra-Processor Job Swapping (IAJS) method to enhance the utilisation of processing space. The intra-processor job swapping procedure is shown in Algorithm 6 and Figure 9.

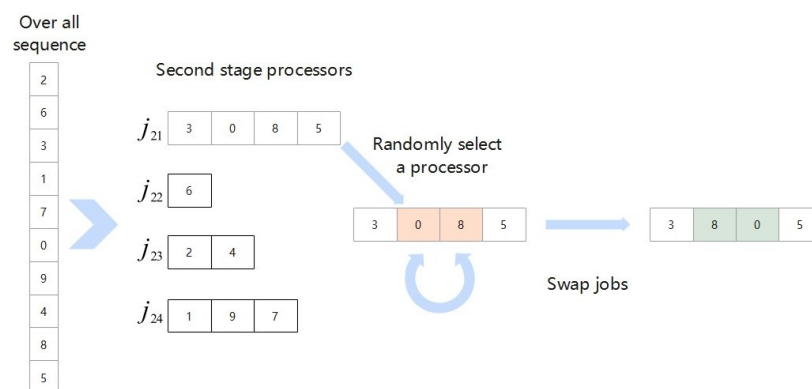
Although the TSMA embeds several domain-specific operators, its two-stage evolutionary framework is broadly applicable; when machining space constraints are ignored, meaning that the spacing requirement and the space of the processor coincide, the TSMA can be applied directly to the classical two-stage HFSP. Analysis of the IPJS and IAJS operators shows that if the scheduling problem includes a parallel machine stage that must satisfy continuous two-dimensional spatial constraints, then both operators can still make a meaningful contribution. Thus, by suitably redefining the problem's constraint set, the TSMA can be adapted to a wide range of other hybrid flow-shop variants.

Algorithm 6 Intra-processor job swapping operation**Require:** Set of processors J at second stage; current schedule of processors π_2 **Ensure:** A new schedule of processors π_2 at second stage if the total tardiness does not increase

```

1:  $j \leftarrow$  A randomly processor in second stage
2:  $T \leftarrow \text{decode2}(\pi_{2j})$ 
3:  $k \leftarrow$  Randomly remove a job from  $\pi_{2j}$ 
4:  $\pi'_{2j} \leftarrow$  Insert  $k$  reinserted into another position in  $\pi_{2j}$ .
5:  $T' \leftarrow \text{decode2}(\pi'_{2j})$ ,
6:  $\Delta T \leftarrow T - T'$ 
7: if  $\Delta T \leq 0$  then
8:    $\pi_{2j} \leftarrow \pi'_{2j}$ 
9: end if
return  $\pi_2$ 

```

**Figure 9.** IAJS operation.**4. Computational Comparison and Statistical Analysis**

In this section, we analyse the factors influencing tardiness in the HFSP-2D using a testbed based constructed on the basis of these factors. Using this testbed, we evaluate the solution efficiency of the NEHedd-2D algorithm under different tie-breaking conditions. The solution quality of the TSMA is compared to that of state-of-the-art algorithms on the testbed and further benchmarked against results from the CPLEX solver. Experimental results demonstrate that the TSMA achieves superior solutions at significantly lower computational costs in all cases.

4.1. Computational Environment

All algorithms were compared under the same conditions, meaning the same computer (Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10 GHz and 60 GB RAM) and the same programming language (C++).

4.2. Testbed

To comprehensively and rigorously validate the effectiveness of the compared algorithms and facilitate future research, we developed a standardised testbed. In this testbed, access constraints between satellites and gateway stations are simplified to enable more focused benchmarking of different algorithms' performance.

4.2.1. Analysis of Influencing Factors

In the problem studied in this paper, the parameters of each instance must be defined, including the number of jobs v , processors (J_i) in stage i , processing times (p_{ik}), and resource requirements (r_{ik}). Fernández-Viagas et al. analysed the impact of job due date

settings on HFSP-type problems [29]. This influence persists in the HFSP-2D while further incorporating job space requirement settings, as manifested through four key properties:

- When $d_j > \max(c_{ik}), \forall i \in I, k \in K$, any solution is an optimal solution.
- When $d_j < \sum_{i=1}^s p_{ij}, \forall j$, the objective shifts from $\min T$ to $\min C_{max}$.
- When $r_{ik} + r_{il} > \omega_{ij}, \forall i \in I, j \in J_i, k, l \in K$, the problem reduces to a two-stage HFSP.
- When $\sum_{k=1}^v r_{ik} < \omega_{ij}, \forall i \in I, j \in J_i, k, l \in K$ (indicating sufficient gateway station bandwidth), the second-stage scheduling becomes unnecessary, reducing the problem to a parallel machine scheduling problem.

This analysis demonstrates that testbed generation must holistically consider the following:

- Relationships between due dates and processing times.
- Relationships between job space requirements and processor capacity constraints.

4.2.2. Due Date Parameter Design

In this paper, the due date parameters are generated using Equations (13)–(15), where T_1 and R_1 are related to the mean and variance of the due dates, respectively, according to a uniform distribution between $\rho_1 \cdot (1 - T_1 - R_1/2)$ and $\rho_1 \cdot (1 - T_1 + R_1/2)$, where ρ_1 is the lower bound for the makespan.

$$T_1 = 0.6 \quad (13)$$

$$R_1 = 0.1 \quad (14)$$

$$\rho_1 = \sum_{i=1}^m \left(\frac{1}{n_1} \sum_{j=1}^{n_1} p_{ij} + \frac{1}{n_2} \sum_{k=1}^{n_2} p_{ik} \right) \quad (15)$$

4.2.3. Space Requirement Parameter Design

In this paper, the space requirement parameters are generated using Equations (16)–(18), where T_2 and R_2 are related to the mean and variance of the resource, respectively, according to a uniform distribution between $\rho_2 \cdot (1 - T_2 - R_2/2)$ and $\rho_2 \cdot (1 - T_2 + R_2/2)$, where ρ_2 is a lower bound for the resource.

$$T_2 = 0.5 \quad (16)$$

$$R_2 = 0.5 \quad (17)$$

$$\rho_2 = \max(\omega_{2j}), \forall j \in J_2 \quad (18)$$

4.2.4. Problem Size Parameter Design

In this paper, the problem size parameters are as presented in Equations (19)–(21).

$$J_1 = 25 \quad (19)$$

$$J_2 = 5 \quad (20)$$

$$v = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\} \quad (21)$$

Our carefully designed testbed minimizes the risk of overfitting by creating a highly diversified evaluation environment. In particular, the due date and space requirement parameters are not fixed; rather, they are sampled from uniform distributions with means and variances chosen to emulate a broad spectrum of time constraints and highly volatile resource demand scenarios. This forces the algorithm to adapt to a variety of constraints rather than memorize a specific pattern. Moreover, the problem size parameter v is systematically scaled over a ten-fold range, enabling a rigorous assessment of the algorithm's scalability across different levels of complexity.

4.2.5. Performance Indicators

In addition, following Fernandez-Viagas and Framinan [1] and Fernandez-Viagas et al. [30], the computational effort of heuristics is evaluated using the Average Relative Percentage Deviation (ARPD). The ARPD is defined in Equations (22) and (23):

$$ARPD_h = \frac{1}{I} \sum_{i=1}^I RPD_{ih}, \forall h = 1, \dots, H \quad (22)$$

$$RPD_{ih} = \frac{OF_{ih} - Best_i}{Best_i} \cdot 100, \forall h = 1, \dots, H \quad (23)$$

where H is the number of algorithms, I is the number of instances, OF_{ih} is the total tardiness of algorithm h for instance i , and $Best_i$ is the minimum total tardiness achieved by any algorithm for instance i .

4.3. Analysis of Satellite–Gateway Access Constraints

To streamline the experimental design, we introduced a simplified model of satellite gateway–access constraints. To validate this simplification, we carried out an access experiment between a single ground gateway station with geographic coordinates in latitude and longitude of $(0^\circ, 0^\circ)$ and a communication satellite constellation. The constellation followed the Walker configuration detailed in Table 3, and the simulation covered 24 h.

Table 3. Walker constellation parameters.

Category	Parameter	Value
Orbital Elements of the Seed Satellite	Semi-major axis	7178.14 km
	Eccentricity	0
	Inclination	45°
	Argument of perigee	0°
	RAAN	0°
	True anomaly	0°
Walker Constellation Layout	Number of orbital planes	6
	Satellites per plane	4
	Inter-plane phasing	Uniform
	Intra-plane phasing	Uniform

As depicted in Figure 10 and Table 4, with a 24 satellite constellation, the ground station enjoyed uninterrupted coverage—access to at least one communications satellite was available 100% of the time, while access to two or more satellites occurred during 28.74% of the total period. These results demonstrate that the simplifications adopted in our testbed for satellite–gateway access constraints are well founded.

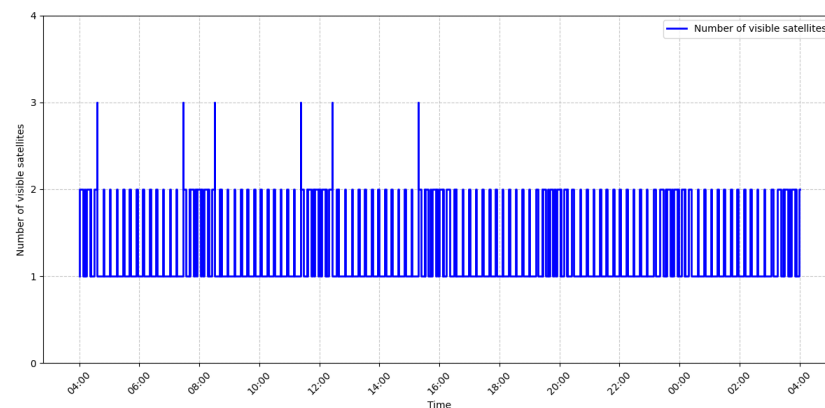


Figure 10. Changes in the number of satellites visible to the gateway station over time.

Table 4. Visibility statistics for different numbers of satellites.

Number of Visible Satellites	Time (s)	Percentage (%)
1	61,564.50	71.26
2	24,773.52	28.67
3	61.98	0.07
Total	86,400.00	100.00

4.4. Parameter Calibration

In this section, we calibrate the parameters of the TSMA using the Taguchi approach. The proposed TSMA consists of three parameters: the local search rate (r_1), crossover rate (r_2), and mutation rate (r_3). An L16 (4^3) orthogonal array was selected for the experimental design, which can accommodate three parameters (r_1, r_2, r_3) with four levels each (Table 5). This design generated sixteen distinct parameter combinations, covering all parameter-level cross-testing while minimising experimental runs. The ARPD values served as the response variable.

Table 5. Parameter values of each factor level.

Parameter	Factor Level			
	1	2	3	4
r_1	0.1	0.2	0.3	0.4
r_2	0.6	0.7	0.8	0.9
r_3	0.1	0.2	0.3	0.4

Through main effects analysis, the average ARPD values across different levels of each parameter are calculated as the response variable values, thereby determining the degree of influence each parameter has on algorithm performance. Figure 11 illustrates the factor level trend of the three critical parameters. Table 6 describes the Average Response Variable (ARV) and delta values at different levels of each parameter.

In summary, the parameters of the TSMA are set to $r_1 = 0.4$, $r_2 = 0.8$, $r_3 = 0.2$.

Table 6. Orthogonal Array and Response Variable (ARV) values.

No.	r_1	r_2	r_3	ARV (%)
1	0.1	0.6	0.1	6.945958
2	0.1	0.7	0.2	4.213748
3	0.1	0.8	0.3	3.512158
4	0.1	0.9	0.4	5.433704
5	0.2	0.6	0.2	4.833371
6	0.2	0.7	0.1	4.915886
7	0.2	0.8	0.4	2.882959
8	0.2	0.9	0.3	4.799043
9	0.3	0.6	0.3	4.241184
10	0.3	0.7	0.4	3.849483
11	0.3	0.8	0.1	3.691779
12	0.3	0.9	0.2	3.333150
13	0.4	0.6	0.4	3.780806
14	0.4	0.7	0.3	3.888193
15	0.4	0.8	0.2	1.433352
16	0.4	0.9	0.1	4.539569

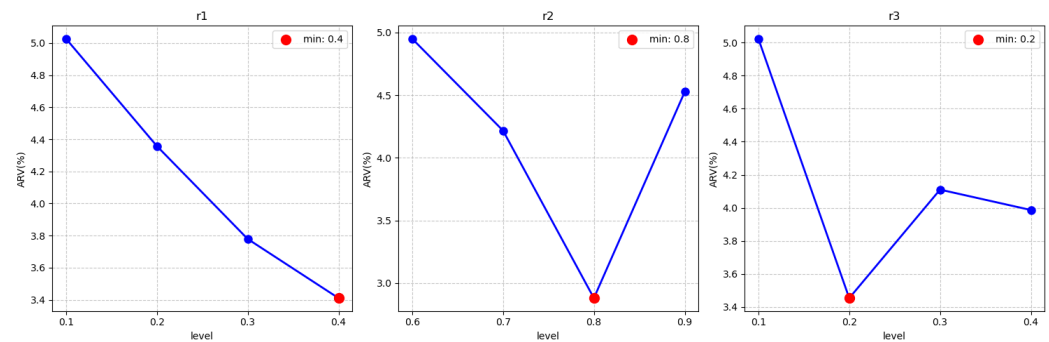


Figure 11. Main effects plot for the TSMA.

4.5. Analysis of TSMA Optimization Dynamics

This section evaluates the solution results for varying numbers of tasks and different iteration counts.

Table 7 shows that the TSMA converges almost instantly on very small instances ($v \leq 20$), where total tardiness stabilizes after only 100–200 generations. For medium-sized problems ($30 \leq v \leq 60$), most improvement is completed within 300–600 generations, and further evolution yields little benefit. On larger instances ($v \geq 70$), the objective value continues to decrease even at 1000 generations, indicating that the search space has not been fully exploited. Relative improvement from generations 100 to 1000 ranges from virtually 0% ($v = 10, 20$) to about 17% ($v = 100$), with conspicuous plateaus at $v = 50$ and 80, where the algorithm appears to fall into local optima. These observations imply that early stopping criteria could safely terminate runs for small and medium problems, whereas larger problems would benefit from more iterations to avoid premature convergence.

Table 7. Total tardiness at different generations for various problem sizes.

Jobs (v)	Generations									
	100	200	300	400	500	600	700	800	900	1000
10	480	480	480	480	480	480	480	480	480	480
20	938	937	937	937	937	937	937	937	937	937
30	914	904	882	882	882	882	882	881	881	881
40	1161	1115	1086	1046	1021	1021	1019	1000	998	998
50	1604	1603	1603	1603	1594	1585	1561	1561	1561	1545
60	1416	1391	1360	1360	1336	1327	1322	1321	1317	1315
70	2165	2098	2041	1989	1905	1876	1874	1874	1873	1868
80	2244	2198	2162	2150	2140	2121	2121	2120	2113	2113
90	1883	1836	1819	1797	1785	1783	1724	1714	1706	1686
100	3288	2894	2854	2815	2807	2769	2755	2729	2729	2728

4.6. Comparative Experiments for Different Algorithms

In this section, the solution results of the proposed TSMA are compared with those of several other algorithms and the MILP model under varying numbers of jobs.

First, the TSMA is compared with the CPLEX solver in terms of solution quality and computational efficiency. IBM ILOG CPLEX version 12.8 is employed for solving the Mixed Integer Linear Programming (MILP) model, with a time limit of 600 s. The average total tardiness measures the solution quality. The results in Table 8 demonstrate that CPLEX only identifies optimal solutions for small-scale instances; its performance deteriorates significantly as instance scales increase.

Table 8. TSMA vs. CPLEX performance in different instances.

Jobs (<i>v</i>)	CPLEX		TSMA	
	Tardiness	CPU Time (s)	Tardiness	CPU Time (s)
10	482	30.06	482	11.31
20	979	600.00	937	22.79
30	1033	600.00	881	25.61
40	1623	600.00	998	26.32
50	-	600.00	1545	31.28
60	-	600.00	1315	31.01
70	-	600.00	1868	39.09
80	-	600.00	2113	44.33
90	-	600.00	1686	84.74
100	-	600.00	2728	81.27

We compared the TSMA with other state-of-the-art algorithms as well: GA [31], HSA [32], and IG [33]. Genetic Algorithms (GAs) are a population-based meta-search technique that operate on a set of solutions simultaneously, whereas Iterated Greedy (IG) and Simulated Annealing (SA) are probabilistic meta-heuristics. Because these algorithms were not originally designed for the HFSP-2D, we adapted them by modifying their neighbourhood search mechanisms, initialization procedures, and encoding–decoding schemes, allowing them to effectively tackle the HFSP-2D instance set. The computational results are reported in Table 9, where the best ARPD value in each group is shown in boldface. Figure 12 depicts a comparison of the ARPD values achieved by the TSMA and the other algorithms, from which the following conclusions are drawn.

The TSMA maintains an ARPD of 0.0% throughout the entire job size range (10–100), showing that its performance is consistently stable and always the best. For smaller problems (10–20 jobs), both the GA and TSMA reach the optimal solution; however, as the size grows, the local search operators embedded in the TSMA reveal a clear advantage. In most settings, the gap between the two algorithms exceeds 5%. This occurs because with more jobs, poor sequencing is more likely to waste processing space on the processors, a situation that the TSMA can effectively avoid.

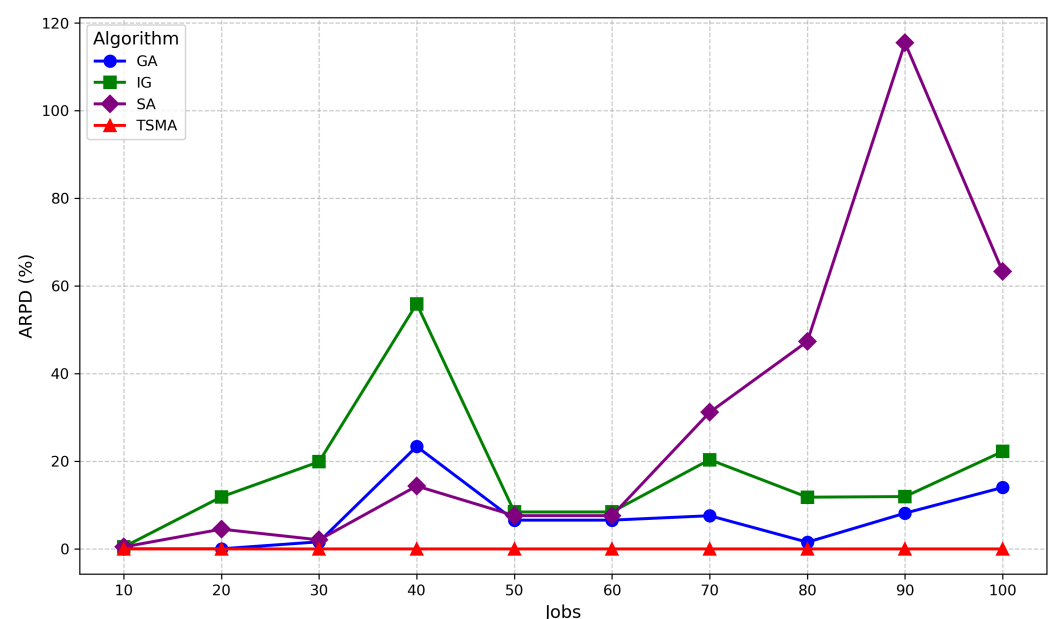
**Figure 12.** Comparison of ARPD values across different algorithms.

Table 9. Comparison of ARPD (%) for the four algorithms at different job sizes.

Jobs (<i>v</i>)	GA	IG	TSMA	SA
10	0.00	0.42	0.00	0.42
20	0.00	11.85	0.00	4.48
30	1.59	19.86	0.00	2.04
40	23.35	55.91	0.00	14.33
50	6.54	8.41	0.00	7.57
60	4.79	20.38	0.00	56.35
70	7.55	20.34	0.00	31.16
80	1.51	11.78	0.00	47.33
90	8.13	11.92	0.00	115.48
100	14.00	22.21	0.00	63.27

4.7. Effectiveness of Local Search Operators

To assess the contribution of the local search mechanism, we performed an ablation study on the TSMA and compared the complete algorithm with three simplified variants:

- W/o IPJS—Keeps IAJS but removes IPJS.
- W/o IAJS—Keeps IPJS but removes IAJS.
- W/o LS—Removes both IAJS and IPJS, i.e., the entire local search component.

The RPD and ARPD results for the four algorithms are plotted in Figure 13. A one-way ANOVA (Table 10) shows a significant main effect of algorithm type on performance ($F(3, 35) = 27.13$, $p < 0.001$). Tukey's HSD post hoc tests (Table 11) further reveal that the full TSMA significantly outperforms every ablated version; discarding IAJS ($p = 0.0403$), IPJS ($p = 0.0111$), or the whole local search module ($p < 0.001$) leads to a substantial performance drop in each case, with the removal of all operations having the most pronounced negative impact (mean difference = 33.56).

No significant difference is detected between the w/o IAJS and w/o IPJS variants ($p = 0.9498$); nevertheless, both outperform the w/o LS version ($p < 0.001$). These results indicate that the local search component is the principal driver of the TSMA's effectiveness.

Table 10. ANOVA results for the different algorithms.

	Sum of Squares	Df	F	<i>p</i>
C (Algorithm)	5698.721	3.0	27.126	3.012161×10^{-09}
Residual	2450.990	35.0	-	-

Table 11. Multiple comparison of means via Tukey's HSD.

Algorithm 1	Algorithm 2	Mean Diff.	<i>p</i> _{adj}	CI Lower	CI Upper
TSMA	w/o IAJS	10.73	0.040	0.36	21.10
TSMA	w/o IPJS	12.73	0.011	2.36	23.10
TSMA	w/o LS	33.56	<0.001	23.19	43.93
w/o IAJS	w/o IPJS	2.00	0.950	−8.09	12.10
w/o IAJS	w/o LS	22.83	<0.001	12.74	32.93
w/o IPJS	w/o LS	20.83	<0.001	10.74	30.92

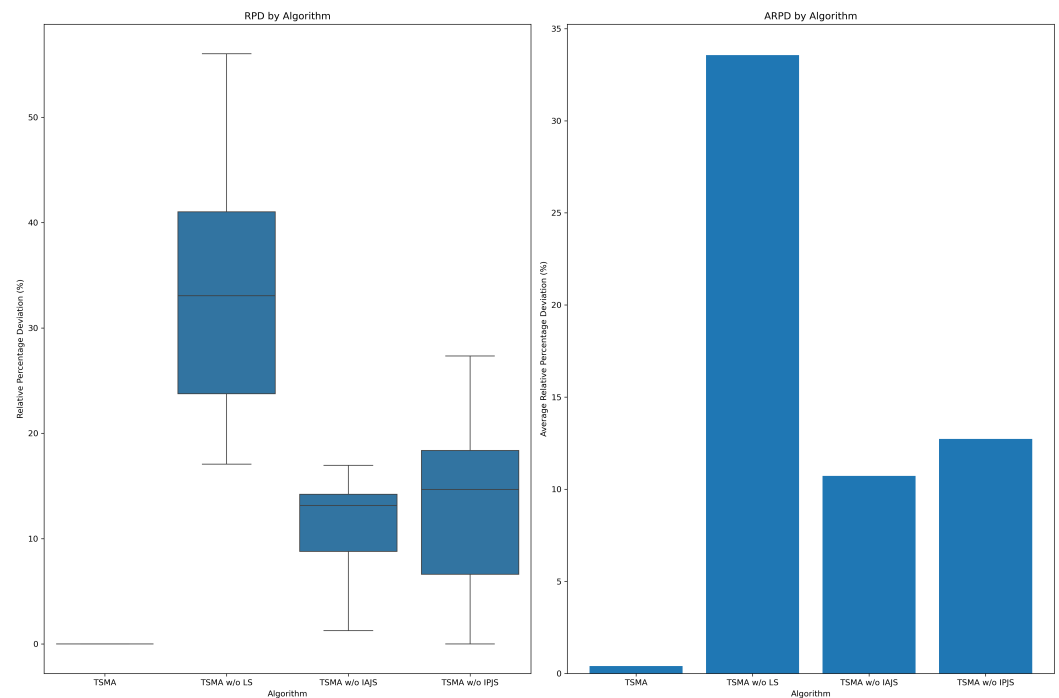


Figure 13. RPD and ARPD results for the different algorithms.

5. Conclusions and Future Work

This study addresses the scheduling of remote sensing data delivery in low-Earth-orbit (LEO) communication constellations, which involves two successive stages: inter-satellite data transfer, and satellite-to-ground downlink. We formalize the problem as a Two-Dimensional Hybrid Flow Shop Scheduling Problem (HFSP-2D) and develop a Mixed-Integer Linear Programming (MILP) model for its exact representation.

To solve the HFSP-2D efficiently, we propose a constructive NEHedd-2D heuristic and a two-stage memetic algorithm which incorporates two problem-specific local search operators. Guided by an analysis of the key problem factors, we designed a dedicated testbed and carried out an extensive campaign of computational experiments. The experimental results confirm the correctness of the MILP model, the effectiveness of the local search operators, and the overall superiority of the TSMA when benchmarked against state-of-the-art algorithms.

Our results show that the proposed TSMA consistently delivers high-quality solutions across all instance sizes. The inclusion of the proposed local search operators significantly enhances the TSMA's performance, while optimal solutions obtained with CPLEX validate the soundness of the MILP formulation.

While the No Free Lunch Theorem precludes universal superiority, the TSMA demonstrates state-of-the-art performance for the HFSP-2D problem class motivated by DTIC scheduling, as evidenced by its consistent 0% ARPD across all tested instances. Its effectiveness stems from the incorporation of components explicitly designed to address the two-dimensional spatial constraints and tardiness minimization objective unique to this domain.

Although the present study has achieved meaningful progress, several issues remain open. We have introduced an MILP formulation capable of solving the HFSP-2D to optimality; however, guided by practical requirements, we have not pursued further research on exact algorithms in this paper. Enhancing the MILP with decomposition techniques such as column generation for the routing decisions offers a promising direction for medium-sized instances, and could extend the applicability of the HFSP-2D model to

other contexts. Moreover, although the proposed TSMA achieves outstanding solution quality in static environments, deploying it in real-time settings remains challenging; schedules must be adjusted swiftly when, for example, a satellite or gateway station fails, or when urgent high-priority tasks suddenly arise. Recomputing the entire schedule from scratch under such conditions is prohibitively time-consuming and fails to satisfy strict real-time constraints. Consequently, our future work will concentrate on developing rapid incremental techniques for schedule repair that can update existing plans within the required time limits.

Author Contributions: Conceptualization, J.Y. and Y.C.; investigation, X.L. and Q.Z.; writing—original draft, J.Y.; writing—review and editing, J.Y. and X.L.; supervision, Y.C.; project administration, J.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the National Natural Science Foundation of China (Grant No. 72271240) and the Natural Science Foundation of Hunan Province (Grant No. 2024JJ4047).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

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

Abbreviations

The following abbreviations are used in this manuscript:

AFSCN	Air Force Satellite Control Network
ARPD	Average Relative Percentage Deviation
ARV	Average Response Variable
BL	Bottom-Left Placement
CPLEX	IBM ILOG CPLEX Optimization Studio
DTIC	Remote Sensing Data Transmission in Communication Constellations
EDD	Earliest Due Date
FAM	First Available Machine
FIFO	First-In, First-Out
GA	Genetic Algorithm
GEO	Geostationary
HFSP	Hybrid Flow Shop Scheduling Problem
HFSP-2D	Hybrid Flow Shop Problem with Two-Dimensional Processor Space
IAJS	Intra-Processor Job-Swapping
IG	Iterated Greedy
IPJS	Inter-Processor Job-Swapping
LEO	Low-Earth-Orbit
MILP	Mixed-Integer Linear Programming
NEH	Nawaz–Enscore–Ham
SA	Simulated Annealing
TSMA	Two-Stage Memetic Algorithm

References

1. Vazquez, A.J.; Erwin, R.S. On the tractability of satellite range scheduling. *Optim. Lett.* **2015**, *9*, 311–327. [[CrossRef](#)]
2. Zhu, W.; Hu, X.; Xia, W.; Jin, P. A two-stage genetic annealing method for integrated Earth observation satellite scheduling problems. *Soft Comput.* **2019**, *23*, 181–196. [[CrossRef](#)]
3. He, L.; Liang, B.; Li, J.; Sheng, M. Joint Observation and Transmission Scheduling in Agile Satellite Networks. *IEEE Trans. Mob. Comput.* **2022**, *21*, 4381–4396. [[CrossRef](#)]

4. Yang, W.; He, L.; Liu, X.; Meng, W.; Chen, Y. A Fast Insertion Tabu Search with Conflict-Avoidance Heuristic for the Multisatellite Multimode Crosslink Scheduling Problem. *Tsinghua Sci. Technol.* **2024**, *29*, 843–862. [\[CrossRef\]](#)
5. Gooley, T.; Borsi, J.; Moore, J. Automating Air Force Satellite control Network (AFSCN) scheduling. *Math. Comput. Model.* **1996**, *24*, 91–101. [\[CrossRef\]](#)
6. Chen, Y.; Lu, J.; He, R.; Ou, J. An Efficient Local Search Heuristic for Earth Observation Satellite Integrated Scheduling. *Appl. Sci.* **2020**, *10*, 5616. [\[CrossRef\]](#)
7. Zhang, J.; Xing, L. An improved genetic algorithm for the integrated satellite imaging and data transmission scheduling problem. *Comput. Oper. Res.* **2022**, *139*, 105626. [\[CrossRef\]](#)
8. Deng, Y.; Yao, F.; Xing, L.; He, L. Inter-satellite data transmission method in satellite network based on hybrid evolutionary algorithm. *Syst. Eng. Electron.* **2023**, *45*, 2931–2940.
9. Chen, X.; Gu, W.; Dai, G.; Xing, L.; Tian, T.; Luo, W.; Cheng, S.; Zhou, M. Data-Driven Collaborative Scheduling Method for Multi-Satellite Data-Transmission. *Tsinghua Sci. Technol.* **2024**, *29*, 1463–1480. [\[CrossRef\]](#)
10. Chen, X.; Li, X.; Wang, X.; Luo, Q.; Wu, G. Task Scheduling Method for Data Relay Satellite Network Considering Breakpoint Transmission. *IEEE Trans. Veh. Technol.* **2021**, *70*, 844–857. [\[CrossRef\]](#)
11. Wu, G.; Luo, Q.; Zhu, Y.; Chen, X.; Feng, Y.; Pedrycz, W. Flexible Task Scheduling in Data Relay Satellite Networks. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 1055–1068. [\[CrossRef\]](#)
12. Li, J.; Wu, G.; Liao, T.; Fan, M.; Mao, X.; Pedrycz, W. Task Scheduling Under a Novel Framework for Data Relay Satellite Network via Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2023**, *72*, 6654–6668. [\[CrossRef\]](#)
13. Yan, Z.; Zhao, K.; Li, W.; Kang, C.; Zheng, J.; Yang, H.; Du, S. Topology Design for GNSSs Under Polling Mechanism Considering Both Inter-Satellite Links and Ground-Satellite Links. *IEEE Trans. Veh. Technol.* **2022**, *71*, 2084–2097. [\[CrossRef\]](#)
14. Rigo, C.A.; Seman, L.O.; Morsch Filho, E.; Camponogara, E.; Bezerra, E.A. MPPT aware task scheduling for nanosatellites using MIP-based ReLU proxy models. *Expert Syst. Appl.* **2023**, *234*, 121022. [\[CrossRef\]](#)
15. Ruiz, R.; Vázquez-Rodríguez, J.A. The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *205*, 1–18. [\[CrossRef\]](#)
16. Choi, H.S.; Lee, D.H. Scheduling algorithms to minimize the number of tardy jobs in two-stage hybrid flow shops. *Comput. Ind. Eng.* **2009**, *56*, 113–120. [\[CrossRef\]](#)
17. Lee, I.S. Minimizing total tardiness for the order scheduling problem. *Int. J. Prod. Econ.* **2013**, *144*, 128–134. [\[CrossRef\]](#)
18. Nawaz, M.; Ensore, E.E.; Ham, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* **1983**, *11*, 91–95. [\[CrossRef\]](#)
19. Lei, D.; Wang, T. Solving distributed two-stage hybrid flowshop scheduling using a shuffled frog-leaping algorithm with memplex grouping. *Eng. Optim.* **2020**, *52*, 1461–1474. [\[CrossRef\]](#)
20. Lu, C.; Zheng, J.; Yin, L.; Wang, R. An improved iterated greedy algorithm for the distributed hybrid flowshop scheduling problem. *Eng. Optim.* **2024**, *56*, 792–810. [\[CrossRef\]](#)
21. Jemmali, M.; Hidri, L.; Alourani, A. Two-stage hybrid flowshop scheduling problem with independent setup times. *Eng. Optim.* **2022**, *21*, 5–16. [\[CrossRef\]](#)
22. Zheng, J.; Chen, S. A Q-learning multi-objective grey wolf optimizer for the distributed hybrid flowshop scheduling problem. *Eng. Optim.* **2024**, *57*, 2609–2628. [\[CrossRef\]](#)
23. Chen, S.; Zheng, J. A Q-learning grey wolf optimizer for a distributed hybrid flowshop rescheduling problem with urgent job insertion. *J. Appl. Math. Comput.* **2025**, *71*, 3645–3670. [\[CrossRef\]](#)
24. Allahverdi, A.; Ng, C.; Cheng, T.; Kovalyov, M.Y. A survey of scheduling problems with setup times or costs. *Eur. J. Oper. Res.* **2008**, *187*, 985–1032. [\[CrossRef\]](#)
25. Naderi, B.; Ruiz, R.; Zandieh, M. *Algorithms for a Realistic Variant of Flowshop Scheduling*; Elsevier: Amsterdam, The Netherlands, 2010; Volume 37, pp. 236–246. [\[CrossRef\]](#)
26. Lin, X.; Chen, Y.; Xue, J.; Zhang, B.; Chen, Y.; Chen, C. Parallel machine scheduling with job family, release time, and mold availability constraints: Model and two solution approaches. *Memetic Comput.* **2024**, *16*, 355–371. [\[CrossRef\]](#)
27. Li, L.; Du, Y.; Yao, F.; Xu, S.; She, Y. Learning memetic algorithm based on variable population and neighborhood for multi-complex target scheduling of large-scale imaging satellites. *Swarm Evol. Comput.* **2025**, *92*, 101789.
28. Du, Y.; Wang, L.; Xing, L.; Yan, J.; Cai, M. Data-Driven Heuristic Assisted Memetic Algorithm for Efficient Inter-Satellite Link Scheduling in the BeiDou Navigation Satellite System. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1800–1816. [\[CrossRef\]](#)
29. Fernandez-Viagas, V.; Framinan, J.M. NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness. *Comput. Oper. Res.* **2015**, *60*, 27–36. [\[CrossRef\]](#)
30. Fernandez-Viagas, V.; Molina-Pariente, J.M.; Framinan, J.M. New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics. *Expert Syst. Appl.* **2018**, *114*, 345–356. [\[CrossRef\]](#)
31. Ishibuchi, H.; Murata, T. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Trans. Syst. Man Cybern. Part C* **1998**, *28*, 392–403. [\[CrossRef\]](#)

32. Naderi, B.; Zandieh, M.; Roshanaei, V. Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness. *Int. J. Adv. Manuf. Technol.* **2009**, *41*, 1186–1198. [[CrossRef](#)]
33. Ruiz, R.; Stutzle, T. An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *Eur. J. Oper. Res.* **2008**, *187*, 1143–1159. [[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.