An adaptive large neighborhood search algorithm for parallel assembly lines scheduling problem with complex fixture constraints

Zhaofang Mao, Yida Xu, Kan Fang, Chengbo Wang, Dian Huang

Abstract

In recent years, the value chain facilitates the transition of competition among enterprises from a single link to a comprehensive one, thereby driving intelligent upgrades within manufacturing enterprises. The intelligent upgrading also imposes new constraints on the traditional assembly line production mode. Inspired by the real production practices of company L, a global intelligent terminal manufacturing enterprise headquartered in China, this study addresses the parallel heterogeneous assembly line scheduling problem with fixture constraints (HALSFC). To tackle this challenging problem, we propose a mixed integer linear programming (MILP) model that aims to maximize the number of completed work orders within a specified time. To our best knowledge, this study is among the first attempts to address the HALSFC problem with setups and related work orders. Due to the NP-hardness of the problem, we propose an improved adaptive large neighborhood search algorithm (IALNS) for solving HALSFC. We evaluate both model functionalities and algorithm effectiveness using instances generated based on the real production data of company L. Extensive experimental results demonstrate the effectiveness and efficiency of IALNS compared to MILP, Tabu search algorithm (TS) and genetic algorithm (GA), especially for medium- and large-scale instances. Additionally, the sensitivity analysis of the quality inspection time, the related work orders proportion and the minimum cooling time of fixtures is also conducted.

Keywords: Value chain, Fixture constraints, Parallel assembly lines scheduling, Metaheuristic

Preprint submitted to Computers & Industrial Engineering

1. Introduction

In recent years, many advanced interconnected technologies and concepts have emerged, including intelligent data-driven methodologies and theories, cloud computing, big data, and value chain digital ecosystems (Bressanin et al., 2021; Priyadarshini et al., 2022). These advancements have significantly accelerated manufacturer's production efficiency and labor output (Zhou et al., 2018). At the same time, the dual carbon policy proposal, the impact of the pandemic, and the complexity of the international landscape have placed manufacturers in a critical crisis necessitating them to vigorously promote value chain digital ecosystem construction and data-driven transformations (Xiufan and Decheng, 2023). Consequently, there is an urgent need to explore strategies for enhancing manufacturers efficiency while reducing costs (Lee and Chuah, 2001).

In the exploration of intelligent transformation, the automation upgrading of assembly lines has emerged as an ongoing trend (Chen et al., 2022). The assembly line, a typically serial production system, significantly enhances production efficiency (Boysen et al., 2022). Although there are certain differences between assembly lines and individual machines, when the assembly line operates continuously without interruptions, it can be approximated as a single non-interruptible machine (Ozbakir et al., 2011). In this production mode, all work orders are divided into small batch containing a specific number of products based on model and delivery date. These work orders are allocated to different assembly lines and processed continuously until completion. Each product's processing flow is continuous and indivisible, as depicted in Figure 1. We consider the machines required for work $J1$–$J6$ in the assembly line processing mode as a unified machine $MS1$, disregarding inter-process influences but focusing on input-output relationships and total processing time. This transformation extends the original parallel machine scheduling problem (PMSP) to a parallel assembly line scheduling problem (PALSP).

In real manufacturing environment, the efficiency of the assembly line processing mode is influenced by various resource constraints, such as tools, workers, and materials (Brucker et al., 1999). Fixtures are widely used in the industry as auxiliary devices to enhance production efficiency (Hunter et al., 2006), but they also impose certain limitations and restrictions on the production process (Hunter et al., 2006). Fixtures share similarities with other production resources such as workers or tools, while they also exhibit distinct differences in terms of working mode and replacement operation (Da Silva et al., 2014). Consequently, addressing complex fixture constraints in parallel assembly line scheduling problem poses more intricate challenges

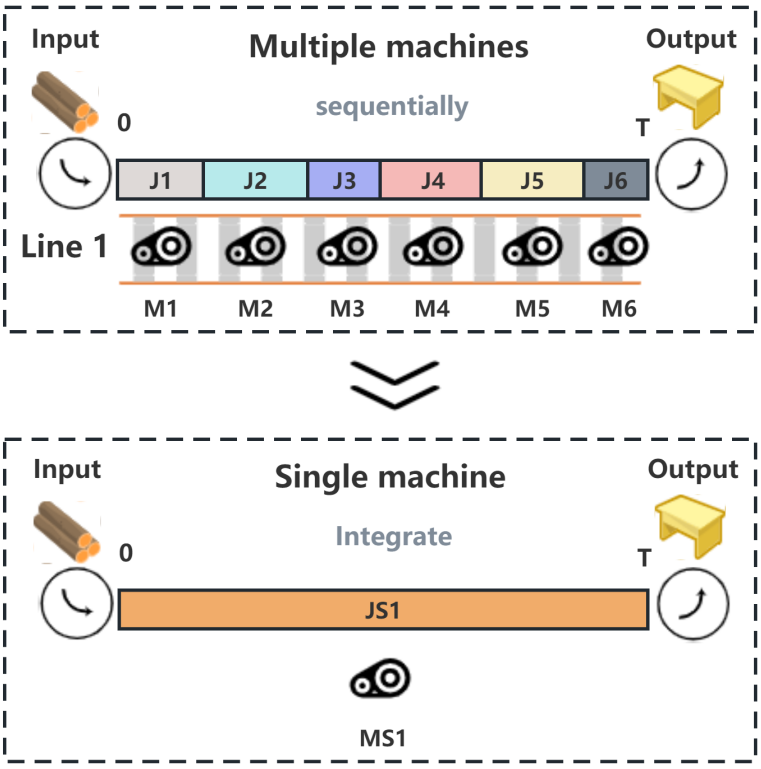

Figure 1: Schematic diagram of assembly line mode and machine production.

compared to general parallel machine scheduling problems with resource constraints, thereby necessitating advanced modeling techniques and algorithmic approaches.

In this study, inspired by the real production scenario of Company L [1], a global intelligent terminal manufacturing enterprise headquartered in China, we propose parallel heterogeneous assembly line scheduling problem with fixture constraints (HALSFC). Similar to previous studies, we adhere to the general constraints of PMSP, ensuring that preemptive jobs are not allowed and jobs are processed continuously, e.g., constraints 2–8 of Section 3.3. Moreover, the pro-
cessing speed of the work order varies on different assembly lines, e.g., p_{ij} for assembly line i and work order j, which is referenced from the studies by Fanjul-Peyro et al. (2017); Fleszar and Hindi (2018); Wang and Alidaee (2019). Specific constraints arising from fixture usage requirements are proposed for depicting the production scenario of company L. The resource constraints imposed by fixtures differ from those imposed by other production resources, such

[1]Actual name of the company is not disclosed to protect confidentiality.

as workers.

Fixture can only be utilized for work orders containing specific model of products, with each fixture having its own maximum continuous use time limit, e.g., constraints 9 and constraints 14–19 of Section 3.3. In order to characterize these constraints, we refer to the relevant constraints on workers' working time and rest time restrictions in Fang et al. (2021) and Fanjul-Peyro et al. (2017). Besides, setup time between different product models are also taken into account (Allahverdi, 2015; Vallada and Ruiz, 2011; Jiang and Wang, 2019). When Company L produces a new product model, a specific quality inspection time I is used to check the small-batch work order, and then arrange the production of the large-batch work order after I time to ensure that there is no quality problem. Company L defines them as related work orders, thus we also introduce this definition into our research. To tackle the proposed HALSFC, we formulate a new mixed integer linear programming (MILP) model and propose an improved adaptive large neighborhood search algorithm (IALNS).

The main contributions of this study are as follows: (i) To the best of our knowledge, this study presents one of the initial attempts to address the HALSFC problem incorporating setup time and related work orders. (ii) A new mixed integer linear programming model is proposed to address this problem that aims to maximize the number of completed work orders within a specified time. (iii) An improved adaptive large neighborhood search algorithm with hybrid acceptance criterion is developed to solve the problem. (iv) A novel strategy with minimum product quantity of the work orders first allocating (MF) is proposed to improve the quality of the initial solution. (v) The performance of MILP and IALNS are verified by extensive experiments, in which different-scale of instances are generated based on the real production data of Company L.

The remainder of the paper is organized as follows. Section 2 reviews the related literature. Problem description and model formulation are presented in Section 3. In Section 4, details of the proposed adaptive large neighborhood search algorithm are given. The computational experiments are conducted in Section 5. In Section 6, we conclude the research and discuss promising future research directions.

2. Literature review

In this section, we review the relevant studies in three streams: (i) parallel machine scheduling problem; (ii) scheduling problem with resource constraints; (iii) application of adaptive large neighborhood search algorithm in the scheduling problem

2.1. Parallel machine scheduling problem

Parallel machine scheduling problem, as a significant research branch in the field of schedul-ing, has received extensive attention from scholars (Tian et al., 2014; Allahverdi, 2016). Vari-ous models and methods have been employed to address PMSP(Zhou et al., 2013). Rajkanth et al. (2017) explore the parallel machines scheduling problem to minimize job completion time. Iterated local search (ILS) and grasp models are developed by Abu-Marrul et al. (2021), which incorporate constructive heuristics and MIP-based neighborhood search methods for solving PMSP. Bitar et al. (2021) consider three different objective functions of PMSP, includ-ing the number of products completed within a given time range, weighted sum of completion time, and number of auxiliary resources moved. Saraç and Tutumlu (2022) present a multi-objective PMSP and use epsilon constraint method to transform them into single objective version. Wang et al. (2023a) investigate a variant of PMSP with multiple time windows and propose two mixed-integer linear programming formulations based on the concept of non-immediate precedence. Santoro and Junqueira (2023) address the PMSP with calendar and present a mixed integer linear programming models for the problem with the aim of minimiz-ing the makespan.

Among the studies conducted on PMSP, a majority of them primarily focus on address-ing the unrelated parallel machine scheduling problem (UPMSP). In this context, each ma-chine's processing and utilization are considered to be mutually independent, ensuring that their respective processing procedures remain unaffected by one another (Sels et al., 2015). Various heuristics have been employed to tackle the UPMSP problem effectively, such as sim-ulation annealing algorithm (Kim et al., 2002), greedy-based fixing algorithm (Fanjul-Peyro et al., 2017), genetic algorithm (Vallada and Ruiz, 2011), local search algorithm (Glass et al., 1994), threshold-accepting method (Chen and Wu, 2006), artificial bee colony algorithm (Lei et al., 2021), iterated hybrid metaheuristic(Lin and Hsieh, 2014; Lopez-Esteve et al., 2023), tabu search algorithm(Lee et al., 2013) and variable neighborhood search(Maecker et al., 2023; Wang et al., 2023b). In contrast, limited research has been conducted on the devel-oping of exact algorithms for solving UPMSP. The existing approaches include a mathemati-cal programming-based algorithm (Fanjul-Peyro et al., 2019) and a cutting plane algorithm (Mokotoff and Chrétienne, 2002).

2.2. Scheduling problem with resource constraints

Researchers have conducted various investigations on the resource constrained project scheduling problem (RCPSP) across various categories of production resources(Herroelen et al.,

1998). According to a survey by Pellerin et al. (2020), RCPSP is a prevalent scheduling problem with extensive applications in manufacturing, production planning, project management, and other domains. There are different types of resources, including workers, production materials, tools, etc., where resource durability, quantity, and characteristics collectively impact the overall production process efficiency(Sánchez et al., 2022).

Production resources, narrowly defined as consumables such as raw materials and energy required for production (Herroelen, 1972). To address this resource constrained project scheduling problem, Yunusoglu and Topaloglu Yildiz (2022) develop an accurate solution method based on constraint programming (CP) and enrich the model by adding lower bound and redundant constraints. Zhigang et al. (2007) establish a general scheduling model for multi-resource constrained job shop scheduling problems and propose a job optimization scheduling algorithm based on ant colony optimization algorithms. Cheng and Lin (2017) develop a Johnson algorithm to solve the two-machine process scheduling problem under resource constraints. Prilutskii (2007) propose a direct search approach for solving resource scheduling problems under multiple criteria. Moreover, Ma et al. (2016) present a project scheduling model based on uncertainty theory that considers uncertain resource-constrained project scheduling problems (URCPSP). Herr and Goel (2016) address a single machine scheduling problem where each job belongs to a specific family, and setups are required between jobs from different families. The authors provide a mathematical formulation and propose a heuristic solution approach for two variations of this problem. Fang et al. (2013) investigate the flow shop scheduling problem with constraints on peak power consumption and validate the proposed model using data from cast iron plate manufacturing.

In a broader sense, production resources encompass not only raw materials but also non-consumable resources such as space, workers, and tools (Herroelen et al., 1998). The RCPSP is addressed by Ranjbar and Kianfar (2010) using genetic algorithm with a novel crossover operator for scheduling activities. For the parallel machine scheduling problem that restricts employee work and rest time, Fang et al. (2021) implement two distinct algorithms based on decomposition and heuristic methods. To address the job shop scheduling problem with stochastic processing time under dual resource constraints of machines and workers, Xiao et al. (2021a) establish a robust job shop scheduling model using a robust scheduling approach. Tao and Hao (2009) conduct a classified scheduling based on resource constraints such as machine tool failure repair time, workers' departure time and the number of tasks included in cancelling orders, and utilize the algorithm combining genetic algorithm and simulated annealing algorithm to obtain the scheduling scheme. Pinto et al. (2013) develop a branch

and bound algorithm to solve the collaborative resource scheduling problem involving workers, tools, etc. A constrained programming model aimed at solving scheduling and tool allocation problems in parallel machine environments is proposed by Gökgür et al. (2018).

2.3. Application of adaptive large neighborhood search algorithm in the scheduling problem

The adaptive large neighborhood search algorithm, an extension of the large neighborhood search (LNS), has emerged as a relatively new metaheuristic algorithm (Ropke and Pisinger, 2006a,b). ALNS has demonstrate remarkable problem-solving capabilities across diverse domains and has been successfully applied in various scenarios (Mara et al., 2022). For instance, Kiefer et al. (2017) design an ALNS algorithm to allocate lectures to different time slots and rooms while avoiding course overlap within the same schedule. Wang et al. (2012) present an ALNS approach for scheduling n jobs on m different parallel machines with multiple time windows. A multi-objective mixed integer linear programming model is proposed by Zeng and Zhang (2021) for single batch processor scheduling, which introduces a two-dimensional rectangular packing constraint. Additionally, ALNS is also used in other routing problems, such as cumulative capacity vehicle routing problem (Ribeiro and Laporte, 2012), static dial-a-ride problem with ride and waiting time (Pfeiffer and Schulz, 2022), electric vehicle scheduling problem (Wen et al., 2016).

To visually illustrate the distinction between our study and previous research, we present a literature comparison table in Table 1. It is evident that there have been extensive investigations on PMSP, but none have addressed the issue with complex fixture constraints. In order to bridge this gap, this study first attempts to tackle parallel assembly lines scheduling problem with complex fixture constraints, in which fixture allocation, worker order assignment, assembly line assignment and setup time are considered. We formulate a MILP model for this problem and derive new lower and upper bounds. Given the complexity of this problem and the proven efficacy of ALNS in solving scheduling problems, we develop an improved adaptive large neighborhood search algorithm to effectively address the problem.

Table 1: Relevant studies on parallel machine scheduling problem.

References	Problem characteristics	Specific constraints	Objective function	Solution methodologies
França et al. (1994)	$IPMSP$	Independent jobs	$\min C_{\max}$	Composite heuristic
Sivrikaya-Şerifoğlu and Ulusoy (1999)	$UPMSP$	Non-preemptive, sequence-dependent setups	$\min ETP$	GA
Weng et al. (2001)	$UPMSP$	Setups, independent jobs	$\min C_{\max}$	Heuristic
Yalaoui and Chu (2002)	$IPMSP$	Non-preemptive, dominance properties	$\min TT$	$B\&B$
Mokotoff (2004)	$IPMSP$	Deterministic, Non-preemptive	$\min C_{\max}$	EA
Dell'Amico et al. (2008)	$IPMSP$	Associated processing times	$\min C_{\max}$	Metaheuristic and EA
Recalde et al. (2010)	$RPMSP$	Restricted processing, eligibility constraints	$\min C_{\max}$	LS
Fang and Lin (2013)	$UPMSP$	Adjustable speed, energy consumption constraints	$\min TT$,$\min PC$	PSO
Hashemian et al. (2014)	$UPMSP$	Resumable jobs, nonavailability periods	$\min C_{\max}$	IE
Afzalirad and Rezaeian (2016)	$UPMSP$	Setups, machine eligibility,precedence	$\min C_{\max}$	GA
Fanjul-Peyro et al. (2017)	$UPMSP$	Additional resources	$\min C_{\max}$	Matheuristic
Wang et al. (2017)	$UPMSP$	Bounded power demand peak, Electricity load limit	$\min C_{\max}$	Two-stage heuristic
Fleszar and Hindi (2018)	$UPMSP$	Renewable resource constraint	$\min C_{\max}$	CP
Yepes-Borrero et al. (2021)	$UPMSP$	Additional resources, sequence dependent setups	(i)$\min C_{\max}$ (ii)$\min NR$	Iterated greedy
Xiao et al. (2021b)	$IPMSP$	Power consumption, energy savings	(i)$\min C_{\max}$ (ii)$\max SV$	$B\&B$
Wang et al. (2022)	$UPMSP$	Controllable processing times, sequence-dependent setup times	(i)$\max N_{cj}$ (ii)$\min C_{\max}$	$LBBD$
Lopez-Esteve et al. (2023)	$UPMSP$	Setups, Additional resources	$\min C_{\max}$	$GRASP$
Wang et al. (2023a)	$UPMSP$	Multiple time windows	$\max TP$	$BRHP$
Maecker et al. (2023)	$UPMSP$	Delivery times, eligibility	$\min TT$	VNS
Wang et al. (2023b)	$UPMSP$	General position-based deterioration	$\min C_{\max}$	$IHVNS$
Santoro and Junqueira (2023)	$UPMSP$	Generic machine availability, eligibility	$\min C_{\max}$	$Gurobi$
This study	$UPALSP$	Setups, fixture constraints, related work orders	$\max N_{cwo}$	$IALNS$

Note: Problem characteristics: $IPMSP$: Identical parallel machine scheduling problem, $UPMSP$: Unrelated parallel machine scheduling problem, $RPMSP$: Related parallel machine scheduling problem, $UPALSP$: Unrelated parallel assembly line scheduling problem; Objective function: $C_{\max}$: The makespan, ETP: The earliness and tardiness penalties, TT: total tardiness, PC: power cost, NR: the number of resources, SV: social value, TP: Total profits, N_{cwo}: number of completed work orders; Solution methodologies: EA: Exact algorithm, LS: Local search, PSO: Particle swarm optimization algorithm, IE: Implicit enumeration algorithm, GA: Generic algorithm, CP: constraint programming method, $B\&B$: Branch-and-bound algorithm, $LBBD$: Logic-based Benders decomposition method, $GRASP$: Greedy Randomized Adaptive Search, $BRHP$: Bidirectional rolling horizon preprocessing algorithm, VNS: Variable neighborhood search algorithm, $IHVNS$: iterative heuristic embedded with a variable neighbourhood search procedure, $IALNS$: Improved adaptive large neighborhood search algorithm

3. Problem description and mathematical formulation

3.1. Problem description

As mentioned earlier, we define the addressed problem as parallel heterogeneous as-sembly line scheduling problem with fixture constraints. In this problem, there are a set $J = \{1, 2, \ldots, n\}$ of work orders, a set $M = \{1, 2, \ldots, m\}$ of assembly lines, a set $L = \{1, 2, \ldots, q\}$ of product models, a set $O = \{1, 2, \ldots, p\}$ of fixtures, a set $F = \{1, 2, \ldots, h\}$ of fixture types. Each work order j, consisting of a quantity of Q_j identical products, has its own delivery time d_j and material balance time r_j. The products' models within a given work order are same while different work orders may have different product models. The work order j is produced non-preemptively on assembly line i with a processing time of p_{ij} depending on the assembly line setting. The production of a work order j requires at most one fixture f and can be started only after the material balance time r_j, and its delivery must occur prior to d_j. When two work orders with different product types are processed successively on a same assembly line, the setups time $s_{ill'}$ needs to be considered, which depends on both the product models involved and the assembly line settings.

Each fixture f has n_f identical copies and all of them are utilized in the same manner. During the use of fixture f, a minimum cooling time CO_f is required when reaching U_f consecutive working-time before the fixture can be utilized again. The continuous processing time and cooling time of each fixture may different. Additionally, there are related pairs of work orders involved in the production of a new product model and the former is a small-batch trial production for this new model. Suppose a related work order pair consists of work order j and k, where $j \neq k$ and $Q_j \leq Q_k$. It is important to note that work order k can only start production after work order j successfully passes the quality check. Thus, the start time S_k of work order k must be equal to or greater than the completion time C_j of work order j, plus the duration time I required for quality inspection, i.e., $C_j + I \leq S_k$.

In the real production scenario, Company L may consider multiple decision-making objectives, including the number of completed work orders within a fixed time, on-time delivery rate, total product output, and so on. Considering the fact that the total number of work orders far exceeds the number that can be scheduled to a shift, as well as the continuous rolling production plan of Company L, we focus on the static scheduling plan within a single shift. Through on-site investigation, the number of completed work orders within a fixed time was identified as the primary delivery indicator for company L. Moreover, work order serves as the smallest and indivisible unit in company L's production process, wherein the quantity of

completed work orders within a fixed time directly determines the extent to which company L can achieve customer satisfaction. Therefore, the objective of the proposed HALSFC is to determine a schedule that maximizes the number of completed work orders within a fixed time T.

3.2. An illustrative example

To intuitively understand HALSFC problem, we provide an illustrative example. There are 14 worker orders with 5 product models, 3 assembly lines and 3 types of fixtures. The corresponding product models and required fixtures are shown in Table 2. Table 3 presents the processing time p_{ij} of each work order on each assembly line. Figure 2 demonstrates a schematic diagram of the scheduling results of the HALSFC problem.

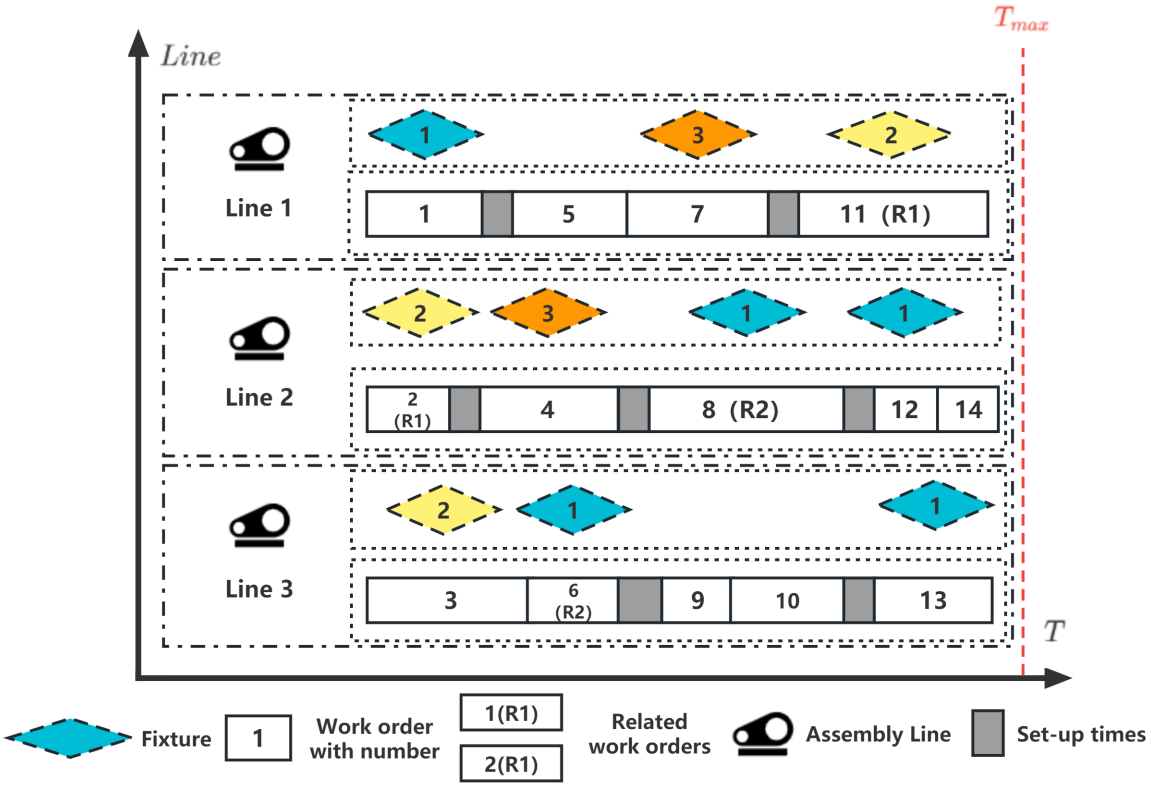

Figure 2: Schematic diagram of result for HALSFC.

Among them, work orders 6 and 8 are related pairs of work orders as well as work orders 2 and 11. The product models in each related pairs are identical and can be processed using

the same type of fixtures. They can be processed on different assembly lines while maintaining a sequential processing relationship. Fixtures may be utilized during the processing of work order, including work orders 1/3/4/7, while others may not be employed, such as work orders 5/9/10. When dealing with work orders involving different product models, it is essential to consider the setup times during continuous processing, which can vary based on assembly line settings and the specific models involved (e.g., worker orders 1 and 5). It is evident that all 14 work orders are successfully processed within total time limit through a well-designed schedule.

Note that when the processing time of a work order crosses T , it is not regarded as a completed work order. Owing to the indivisibility of the work order, we also refrain from considering the processed products of the order. In addition, when one kind of fixtures are all used for processing or cooling, e.g., there are no such kind of free fixtures, the work orders that need to be processed by such kind of fixtures have to wait until a free fixture is released. Once the use or cooling of the fixtures start, it must continue uninterrupted until reaching maximum continuous working time or minimum cooling time, respectively.

Table 2: Product models and corresponding types of fixtures.

J	L	F	J	L	F
1	1	1	8	1	1
2	2	2	9	5	–
3	2	2	10	4	–
4	3	3	11	2	2
5	4	–	12	1	1
6	1	1	13	1	1
7	3	3	14	5	–

Note: J: work order number; L: product models; F: corresponding types of fixtures used to process; —: work orders can be processed without fixtures.

3.3. Mathematical formulation

To begin with, we let T as the length of the planning horizon, and $\mathcal{T} = \{0, 1, 2, \ldots, T-1\}$ represents the corresponding set of discrete time instances obtained by dividing T into uniform segments. Each segment t begins at time instant t and ends at time instant $t+1$. Subsequently, we define the following variables in Table 4:

Table 3: Processing time of work orders for example.

J	p_{ij}		
	$i=1$	$i=2$	$i=3$
1	7	9	13
2	15	4	8
3	15	15	11
4	10	9	10
5	7	12	15
6	3	4	5
7	10	8	15
8	15	11	13
9	17	6	4
10	5	12	7
11	11	15	18
12	5	3	4
13	12	13	11
14	4	3	2

Note: J: work order number; p_{ij}: processing time of work order j on assembly line i.

Considering the above advantages, we formulate the HALSFC problem as the following mixed integer linear programming model. For simplicity, the model is denoted as MILP.

$$\text{maximize} \quad \sum_{\mathrm{t}\in\mathrm{T}}\sum_{\mathrm{i}\in\mathrm{M}}\sum_{\mathrm{j}\in\mathrm{J}}\mathrm{x}_{\mathrm{ijt}} \tag{1}$$

$$\text{s.t.} \quad E_j * r_j \leqslant S_j, \quad \forall j \in J \tag{2}$$

$$C_j \leqslant d_j * E_{\mathrm{j}}, \quad \forall j \in J \tag{3}$$

$$\sum_{i\in M}\sum_{0}^{\mathrm{T}} x_{ijt} \leq 1, \forall j \in J \tag{4}$$

$$\sum_{j\in J}\sum_{s=\max\{0,\mathrm{t}-p_{ij}+1\}}^{\mathrm{t}} x_{ijs} \leq 1, \forall i \in M, t \in \{0,\ldots,T-1\} \tag{5}$$

$$y_{ijt} = \sum_{k=0}^{t} x_{ijk}, \forall i \in M, j \in J, t \in \{1,\ldots,p_{ij}-1\} \tag{6}$$

Table 4: The summary of notations.

Notation	Definition
indices:	
j,k	Work order index, $j,k \in \{1,2,\ldots,n\}$, where n is the number of work orders to be scheduled.
i	Assembly line index, $i \in \{1,2,\ldots,m\}$, where m is the number of assembly lines.
l	Product model index, $l \in \{1,2,\ldots,q\}$, where q is the number of product models.
f	Fixture type index, $f \in \{1,2,\ldots,h\}$, where h is the number of fixture types.
o	Fixture index, $o \in \{1,2,\ldots,p\}$, where p is the number of fixtures.
Parameters:	
J	Set of work orders to be scheduled, $J = \{1,2,\ldots,n\}$
M	Set of assembly lines, $M = \{1,2,\ldots,m\}$
L	Set of Product models, $L = \{1,2,\ldots,q\}$
F	Set of fixture types, $F = \{1,2,\ldots,h\}$
O	Set of fixtures, $O = \{1,2,\ldots,p\}$
R	A sufficiently large positive number.
Q_j	Number of products included in work order j.
r_j	Material balance time for work order j.
d_j	Delivery time for work order j.
p_{ij}	Process time of work order j by assembly line i.
$s_{ill'}$	Setup time from product model l to product model l' on assembly line i.
U_f	Maximum continuous usage time for fixture type f.
CO_f	Minimum cooling time after consecutive working for fixture type f.
I	Interval inspection time.
Decision and auxiliary variables:	
δ_{ijk}	1, if work order j and work order k are processed on assembly line i at the same time, and j is processed before k; 0, otherwise.
x_{ijt}	1, if assembly line i starts processing the work order j at time t; 0, otherwise.
ξ_{ij}	1, if work order j is processed on assembly line i; 0, otherwise.
y_{ijt}	1, if assembly line i is operating work order j on at time t; 0, otherwise.
θ_{ot}	1, if fixture o starts break after finishing some work orders; 0, otherwise.
z_{ot}	1, if fixture o is being used at time t; 0, otherwise.
E_j	1, if work order j is accepted; 0, otherwise.
α_{jl}	1, if the product model contained in work order j is model l; 0, otherwise.
γ_{fo}	1, if the type of fixture f is type o ; 0, otherwise.
β_{lo}	1, if the work order containing product model l requires the use of fixture o for processing; 0, otherwise.
u_{jk}	1, if work order j and work order k are related work orders, j is processed before k; 0, otherwise.
S_j	The start time of work order j.
C_j	The complete time of work order j.

$$y_{ijt} = \sum_{k=t-p_{ij}+1}^{t} x_{ijk}, \quad \forall i \in M, j \in J, t \in \left\{p_{ij}, \ldots, T\right\} \tag{7}$$

$$\sum_{k=t}^{t+p_{ij}-1} y_{ijk} \geq x_{ijt} * p_{ij}, \quad \forall i \in M, j \in J, t \in \left\{1, \ldots, T - p_{ij}\right\} \tag{8}$$

$$\sum_{o \in O} \mathrm{z_{ot}} \cdot \gamma_{fo} \leq n_f, \forall t \in \{0, \ldots, T-1\}, \mathrm{f} \in F \tag{9}$$

$$C_j + I \leq \left(1 - \mathrm{u_{jk}}\right) * R + S_k, \quad \forall j, k \in J \tag{10}$$

$$C_j = S_j + \sum_{i \in M} \xi_{ij} * p_{ij}, \quad \forall j \in J \tag{11}$$

$$S_j \geq C_k + \sum_{l \in L} \sum_{i \in L} \delta_{jk} * s_{ill} * \alpha_{jl} * \alpha_{jl}, \quad \forall i \in M; j, k \in J, j \neq k; l \neq i \tag{12}$$

$$S_j = \sum_{i \in M} \sum_{t \in T} t * x_{ijt}, \quad \forall j \in J \tag{13}$$

$$z_{ot} = \sum_{i \in M} \sum_{j \in J} \sum_{s=\max\left\{0, t-p_{ij}+1\right\}}^{t} \sum_{l \in L} x_{ijs} * \alpha_{jl} \cdot \beta_{lo}, \forall o \in O, t \in \{0, \ldots, T-1\} \tag{14}$$

$$\sum_{s=t}^{t+U_f} \mathrm{z_{os}} \leq U_f + \left(1 - \gamma_{fo}\right) * R, \forall o \in O, \mathrm{f} \in F, t \in \left\{0, \ldots, T - U_f - 1\right\} \tag{15}$$

$$z_{ot} + \theta_{ot} \leq 1, \forall o \in O, \quad t \in \{0, \ldots, T-2\} \tag{16}$$

$$z_{ot} \geq \theta_{o,t+1}, \forall o \in O, \quad t \in \{0, \ldots, T-1\} \tag{17}$$

$$z_{ot} - z_{o,t+1} \leq \theta_{o,t+1}, \forall o \in O, t \in \{0, \ldots, T-2\} \tag{18}$$

$$\left(1 - \gamma_{fo}\right) * R + \sum_{s=t}^{t+C_f-1} (1 - z_{os}) \geq C_f * \theta_{ot}, \forall o \in O, \mathrm{f} \in F, t \in \left\{0, \ldots, T - C_f\right\} \tag{19}$$

$$S_j \geq 0, \forall j \in J \tag{20}$$

$$\sum_{i \in M} \xi_{ij} \leq 1, \quad \forall j \in J \tag{21}$$

$$x_{ijs} = 0, \forall j \in J, i \in M, s \in \left\{T - p_{ij} + 1, \ldots, T\right\} \tag{22}$$

$$\sum_{i \in M} \xi_{ij} = \sum_{i \in M} \sum_{t \in \mathcal{J}} \mathrm{x_{ijt}}, \quad \forall j \in J \tag{23}$$

$$x_{ijt}, y_{ijt}, v_{ijkt}, z_{ot} \in \{0, 1\}, \quad \forall j, k \in J, \quad i \in M, \quad t \in \mathcal{T}, \quad o \in O; \quad j \neq k \tag{24}$$

The objective 1 is to maximize the number of processed work orders within the time limit. Constraints 2-3 are used to indicate the acceptance of a work order for processing. Specifically, work order j can only commence processing after time r_j , and it is considered as a valid completed work order only if it is finished before time d_j . Constraints 4-5 ensure that each work order is processed at most once, and each assembly line can only assemble one work order at any time. Constraints 6-8 define the relationship between variables x_{ijt} and y_{ijt} . Constraints 9 guarantee that the quantity of each type of fixture utilized on all assembly lines at any time does not exceed the upper limit specified for such fixtures.

Constraints 10 define the relationship between related work orders, in which the start time of large-batch work order k should not precede the complete time of small-batch work order j plus the fixed inspection time. Constraints 11 define the relationship between the start time and complete time of a given work order. Constraints 12 define the set-up time of work orders with different product models which are processed in succession on a same assembly line. Constraints 13 define the relationship between S_j and $x_{i\ jt}$. Constraints 14 define the relationship between z_{ot} and x_{ijt} . Constraints 15 ensure that the continuous use time of fixture o should not exceed the maximum continuous use time U_f based on its fixture type f . Constraints 16-18 define the relationship between z_{ot} and θ_{ot} . Constraints 19 ensure the minimum cooling time CO_f after U_f consecutive prossesing time. Constraints 20-22 define the range of variables S_j , $x_{i\ jt}$ and $\xi_{i\ j}$ in certain cases. Constraints 23 define the relationship between x_{ijt} and ξ_{ij} . Constraints 24 present the domains of the decision variables.

3.4. Lower bound and upper bound

In this subsection, we provide a basic lower bound (LB) and upper bound (UB) for the objective of HALSFC problem, i.e., Equation 1.

In the subsequent section, we introduce the IALNS approach for effectively solving HALSFC problem. Considering the fact that HALSFC is a maximum optimization problem, it is obvious that any feasible solution provided by IALNS is a lower bound for HALSFC. In this study, we utilize the initial solution N_{MF}, which is obtained by the minimum product quantity of the work orders first allocating strategy and the greedy-scheduling based approach of IALNS (see Section 4.1 and Section 5.1), as the lower bound for HALSFC, i.e., $LB = N_{MF}$.

To obtain an upper bound, we define a new model by relaxing the constraints 2–3 of the original MILP model, i.e., any work order can be processed at the start time, and the work order that ends processing at any time can be delivered. Obviously, the optimal value of the

new model is an upper bound of HALSFC. Thus, we can obtain a UB by solving the new model utilizing the Gurobi solver.

4. Improved adaptive large neighborhood search algorithm

It is easy to see that when we set $I = 0$, $u_{jk} = 0$, $r_j = 0$, $d_j = +\infty$, $s_{ill'} = 0$, the form of HALSFC problem is quite similar to integrated employee and parallel machine scheduling problem with consecutive working-time constraints (IEMSCW), which is NP-hard in the ordinary sense with $m \geq 2$ (Fang et al., 2021). Therefore, HALSFC problem is also NP-hard, which is difficult to solve especially for medium- and large-size instances.

It is well know that the time cost of exact algorithms in solving NP-hard problems is generally much higher than heuristic algorithms (Akbar et al., 2001), while the solution efficiency is an important requirement for real production scenario (Wauters et al., 2012). Consider-ing NP-hardness of the HALSFC problem and the successful application of the adaptive large neighborhood search algorithm in production scheduling, this study adopts ALNS as a solution approach for the proposed problem(He et al., 2019). To further enhance the performance, we improve the ALNS by incorporating the key advantage of simulated algorithm, i.e., Metropolis acceptance criterion, to expand the neighborhood search effect of the algorithm. In the subsequent subsections, we introduce the details of the improved adaptive large neighborhood search algorithm.

4.1. Initial solution generation

4.1.1. Encoding

Considering the characteristics of HALSFC, such as the related work orders, we encode the solution by proposing a priority list to ensure that small-batch work orders are processed first and there is a minimum interval inspection time I between two related work orders. Work orders are sorted in the descend order of the priority value. Subsequently, work orders are scheduled according to the priority list. Additionally, if two work orders are related work orders, the priorities of them are rearranged to ensure the small-batch work order is processed in advance.

4.1.2. Decoding

As mentioned earlier, there are five decisions in the HALSFC: (i) work order assignment to the fixtures of the corresponding model; (ii) work order assignment to the assembly lines;(iii) determination of the use and rest time of the fixture; (iv) determination of the processing sequence of the work order; (v) determination of the setups. Considering the fact that decisions (iv)–(v) can be easily obtained from the decisions (i)–(iii) and the priority list procedure, we focus on the first three decisions. The decoding strategy is listed as follows:

As presented in Algorithm 4.1, we first initialize the unscheduled work order list J_U , the scheduled work order list J_S , the assembly line occupation list M_L , the fixture occupation list F_L , the priority list P, and the set of fixtures O_j that can be used for the processing of work order j. Then, the work orders are allocated in a greedy manner, considering both the prior-ities and fixture constraints. We establish the termination criterion that no work orders from the unscheduled work order list can be incorporated into the current scheduling status. If the termination condition is not reached, we traverse the unscheduled work order list J_U , the assembly line list M_L and fixture list F_L . At this time, the work order is allocated to the assembly line that finished processing at the earliest, ensuring maximum utilization of each assembly line. The first idle fixture corresponding to the model is selected. After a work order is scheduled, we check whether the currently used fixture can process any work order in J_U . If not, the fixture is set to cooling state and it can only be utilized after a cooling time. It should be noted that the cooling time of the fixture may vary with the product model.

To visualize the decoding procedure, we illustrate the decoding diagram in Figure 3. Note that each priority list corresponds to a feasible solution. Taking advantage of the above encoding and decoding procedures, we can obtain an initial solution by generating the priority list with minimum product quantity. In other words, work orders comprising fewer products are assigned higher priorities and allocated to the assembly lines and the fixtures by the greedy based work order allocation scheme.

Based on HALSFC's problem characteristics, we propose an initial solution generation strategy with minimum product quantity of the work orders first allocating (MF). In the unscheduled work order list, we assign a higher priority to the work orders that include fewer products, so that small-batch work orders can be processed as early as possible, which is more conducive to the increase of the objective function. In addition, we compare the proposed strategy with the other common initial solution generation strategies, e.g., First Come First Serve, FCFS; Earliest Due Date, EDD; Longest Processing Time First, SPT; Random generating,

Algorithm 4.1 Decoding Procedure Based on Greedy Scheduling

Input: J, O, M
Initialize: J_U, J_S, M_L, F_L, P, O_j
if $u_{jk} = 1, Q_j > Q_k$ **then**
  $priority_j < priority_k$
while Need_continue=True **do**
  for $j \in J_U$ **do**
    S=**False**
    for $m \in M$ **do**
      if S=**True then**
        break
      for $o \in O_j$ **do**
        if $C_o < U_f$ **then**
          Schedule work order j for processing on assembly line m by fixture o
          Update: J_U, J_S, M_L, F_L, $S = True$
        else
          Cooling the fixture o for CO_f time
          continue
        Update: C_o, F_L

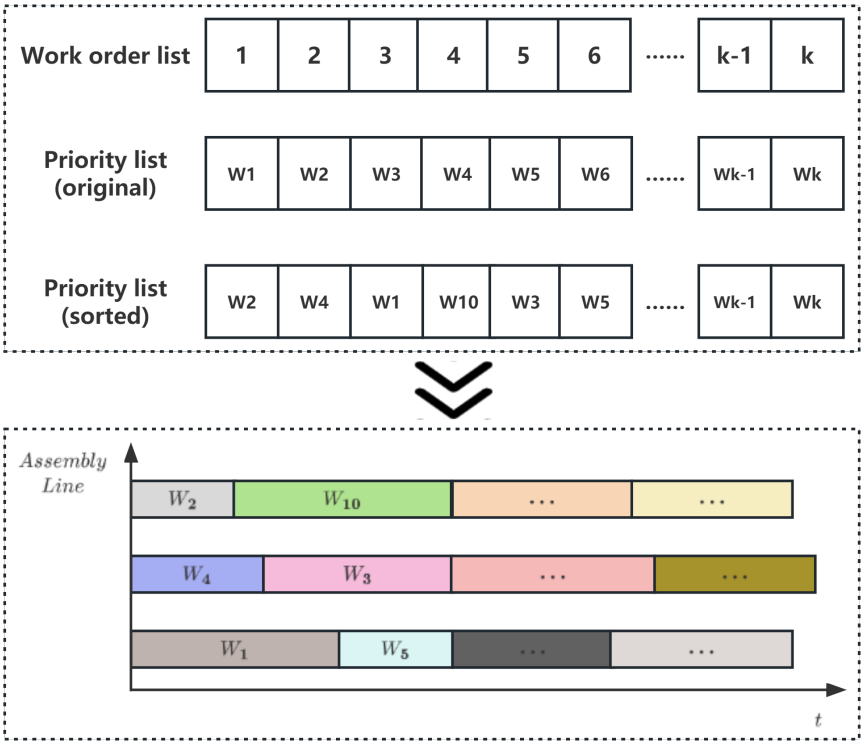

Figure 3: The decoding procedure based on greedy scheduling.

RD. Since the efficiency difference for these techniques are not quite obvious on small-scale instances, we select three medium-scale instances and three large-scale instances used in Section 5.1 for experimental comparison. As shown in Table 5, the result shows that the proposed MF strategy outperforms others in most cases.

Table 5: Comparison of different initial solution generation strategies using IALNS.

Instance	*ML*	*FCFS*	*EDD*	*SPT*	*RD*
J201	80	70	73	78	73
O201	66	62	60	63	58
M201	66	62	67	60	66
J301	222	225	218	220	224
O301	230	220	228	224	222
M301	201	196	189	199	195

Note: The figures in the table represent the number of completed work orders included in the final solution.

4.2. Neighborhood generation

In Section 4.1, we obtain an initial solution. For the process of neighborhood generation, destroy operators and repair operators are commonly utilized (Larranaga et al., 1999). We propose four kinds of destroy operators and two kinds of repair operators referring to the previous study of Liu et al. (2017), and define the set of destroy operators and the set of repair operators as Ω^- and Ω^+, respectively. Figure 4 shows a schematic diagram of these operators. Each destroy operator selects v elements from the priority list of a given solution, removes them, and saves them into the removed set V. The elements in V will be reinserted to the destroyed solution through repair operators to produce a new solution. In each neighborhood transformation operation, operators are selected according to certain weights.

Details of the four destroy operators are as follows:

Random removal: This is the simplest removal operator. Select v work orders from the priority list randomly and remove them.

Priority removal: This is a greedy operator, which removes v work orders with the highest processing time on each assembly line.

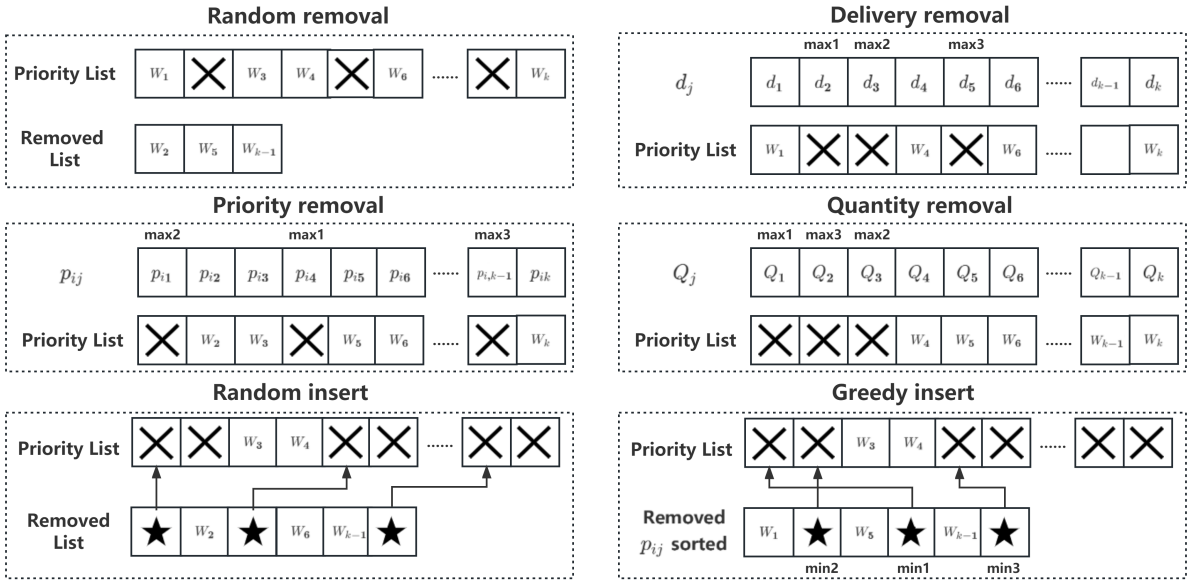

Figure 4: Schematic diagram of destroy operators and repair operators.

Delivery removal: This operator removes the work orders according to the delivery dates, i.e., removes v work orders with the latest delivery dates. In order to optimize the objective function, when removing the work orders with late delivery dates, priority will be given to scheduling work orders with relatively early delivery dates.

Quantitive removal: This operator removes v work orders with the largest number of products. Typically, work orders with fewer products have shorter processing times and are prioritized to maximize the number of completed work orders. Consequently, the v work orders containing the highest quantity of products are sequentially eliminated.

Details of the two repair operators are as follows:

Random repair: This operator is similar to random removal operator. v work orders are randomly selected from the removed set and randomly inserted to the priority list of the destroyed solution.

Greedy repair: This operator arranges the work orders in the removed list based on their processing time and selects v work orders with the shortest processing time to be inserted into the top v available positions of the repaired list.

4.3. Adaptive strategy

In section 4.2, four types of destroy operators and two types of repair operators are proposed. Subsequently, we introduce the adaptive implementation process of IALNS, including operator weight update and acceptance criteria.

4.3.1. Operator weight update

The weight of each destroy operator and repair operator are denoted as ρ_d and ρ_r, respectively. Similarly, we define the score of each destroy operator and repair operator as ψ_d and ψ_r, respectively. We assign different scores to the operators in each iteration process by the quality of the solution, and update the operator weight using the score to ensure that the operator with higher weight is more likely to be selected in the next iteration. During each algorithm iteration, we use Equation 25 to calculate the weight of each operator where $\lambda \in [0,1]$ is the reaction factor.

$$\rho_d = \lambda\rho_d + (1-\lambda)\psi_d, \quad \rho_r = \lambda\rho_r + (1-\lambda)\psi_r \tag{25}$$

Additionally, we use the roulette wheel selection procedure to determine the neighborhood operators, ensuring that the selection probabilities of operators are positively correlated with their weights. After a algorithm iteration is finished, we update the score of the used operators, including both destroy operator and repair operator, according to the quality of the obtained solution.

$$\psi_d, \psi_r = \begin{cases} \epsilon_1 & \text{if the new solution is a new global best} \quad (26) \\ \epsilon_2 & \text{if the new solution is better than the current one} \quad (27) \\ \epsilon_3 & \text{if the new solution is accepted} \quad (28) \\ \epsilon_4 & \text{if the new solution is rejected} \quad (29) \end{cases}$$

We dynamically adjust the scores of the operators based on the quality of new found solution, increasing them when a better solution is discovered and decreasing them if the new solution is of lower quality. This adaptive approach ensures that the entire process remains responsive to changing conditions. The evaluation function of operator score is shown in Equation 26–29, where $\epsilon_1 \geq \epsilon_2 \geq \epsilon_3 \geq \epsilon_4$.

4.3.2. Acceptance criteria

Due to the lack of clear acceptance and termination criteria, the adaptive large neighborhood search algorithm is often combined with the acceptance and termination criteria of other algorithms (Santini et al., 2018). Metropolis acceptance criterion, the key advantage of simulated annealing algorithm, can be used as one of the acceptance criteria of ALNS (Gunawan et al., 2020). In this study, we also adopt this hybrid framework.

During each algorithm iteration, if the objective of the temporary solution $N(t)$ is greater than that of the current solution $N(c)$, the temporary solution is assigned to the current solution. Thereafter, if the objective of the current solution $N(c)$ is greater than that of global best solution $N(b)$, the current solution is assigned to the optimal solution. Otherwise, if the temporary objective is smaller than the current objective, that is, temporary solution is worse than current solution, metropolis acceptance criterion is used to probabilistic includes the temporary inferior solution. We implement the above process by generating a random number rnd and compare it with the acceptance probability, where $rnd \in [0,1]$. The acceptance probability P_c is shown in Equation 30, where T denotes control parameter (temperature).

$$P_c\{\text{ accept } j\} = \begin{cases} 1 & \text{if } N(c) \leqslant N(t), \\ \mathrm{e}^{-((N(c)-N(t))/T)}, & \text{if } N(c) > N(t), \end{cases} \tag{30}$$

$T \in R^{+}$ denotes control parameter (temperature)

Cooling process follows $T_\tau = \alpha T_{\tau-1}$, where $\alpha \in [0,1]$ is the cooling rate. Let P_i and P_f as the acceptance criterion in the initial solution and final solution, respectively. We define the cooling rate as Equation 31 (Bennage and Dhingra, 1995), where $\tau_{\max}$ is the maximum of algorithm iterations.

$$\alpha = \left(\frac{\ln P_i}{\ln P_{\mathrm{f}}}\right)^{1/(\tau_{\max}-1)} \tag{31}$$

Figure 5 shows the whole algorithm search process of a large example, we can see that the calculation result converges in about 60 iterations. Taking the above advantages, we summarize the outline of IALNS in Algorithm 4.2, where T_f denotes termination temperature, T_i denotes initial temperature, x denotes current solution, x^t denotes temporary solution and x^b denotes global optimal solution.

Algorithm 4.2 Improved Adaptive Large Neighborhood Search(IALNS)

Input: A feasible solution x using MF strategy (see Section 4.1.2), parameters of the problem$(\Omega^-, \Omega^+, \tau_{max}, T_f, T_i, \epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4)$
Initialize: $x^b = x; \rho_d = (1 \ldots . 1); \rho_r = (1 \ldots, 1)$
while $\tau < \tau_{max}$ **do**
  while $T > T_f$ **do**
    select destroy and repair methods $d \in \Omega^-$and $r \in \Omega^+$using the roulette strategy based on the scores of the methods ρ_d and ρ_r;
      $x^t = r(d(x))$
    Obtain $N(x), N(x^t)$ and $N\left(x^b\right)$ using the work order allocation scheme in Algorithm 4.2.
    if $N(x) \leq N(x^t)$ **then**
      $x = x^t$
      if $N\left(x^b\right) \leq N(x)$ **then**
        $x^b = x$
        Update: ρ_d, ρ_r using Equation 25 and 26
      else
        Update: ρ_d, ρ_r using Equation 25 and 27
    else
      Generate random number rnd and calculate acceptance criterion P_c using Equation 30
      if $rnd < P_c$ **then**
        $x = x^t$
        Update: ρ_d, ρ_r using Equation 25 and 28
      else
        Update: ρ_d, ρ_r using Equation 25 and 29
    Calculate the cooling rate α at this time using Equation 31
    Update: $T = \alpha T$
  Update: $T = T_i$, $\tau = \tau + 1$
return x^b

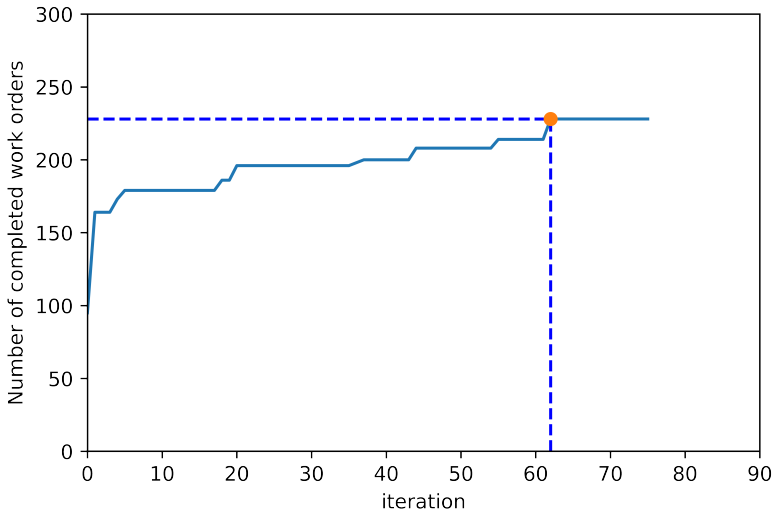

Figure 5: Convergence curve of IALNS for an example.

5. Experimental study

5.1. Experimental setting

As stated earlier, this study is inspired by the practice of company L. However, for the sake of data security and company interests, the actual production data of company L complies with the confidentiality rules. Therefore, on the basis of part real production data provided by Company L, we have generated the experimental data of different scales to validate the effectiveness of the proposed IALNS. Nevertheless, we have consulted extensively with Company L to ensure that our model and algorithm meet the requirements of real-world production scenarios. We generate three scales of instances according to the number of work orders, i.e., 20, 100 and 300 for small-scale instances, medium-scale instance and large-scale instance, respectively. The proportion of related work orders is set to 30% of number of all work orders, for example, if number of work orders is 100, there are 30 related work orders with 15 pairs. Other parameters, such as the number of assembly lines m, the number of copies of each tool n_f, the type of tool f, the product models q, and time window T, are shown in Table 6.

As we know, the relationship between processing time p_{ij} and time window T exerts a discernible impact on the experimental results. In this study, we assume that the processing time of work orders following $(5, 30)$ uniform distribution, which is consistent with the research of Peng and Liu (2004). The setup time of different models $s_{ill'}$ is defined to follow a $(3, 8)$ uniform distribution, as described in the setting of Vallada and Ruiz (2011).

Tabu search algorithm (TS) has demonstrated good performance in solving UPMSP while

Table 6: Parameter settings of instances.

Scenario	I	CO	small instances ($n = 20$)					medium instances ($n = 100$)					large instances ($n = 300$)				
			m	f	n_f	q	T	m	f	n_f	q	T	m	f	n_f	q	T
1	15	(5,7)	2	2	1	2	40	3	2	1	2	300	4	2	1	2	850
2		(5,7,9)	2	3	1	3	50	3	3	1	3	280	4	3	1	3	880
3		(5,7)	3	2	2	2	60	4	2	2	2	320	5	2	2	2	900
4	18	(4,6)	2	2	1	2	40	3	2	1	2	300	4	2	1	2	850
5		(5,6,7)	2	3	1	3	50	2	3	1	3	280	5	3	1	3	880
6		(6,4)	3	2	2	2	60	3	2	2	2	320	4	2	2	2	900
7	20	(4,7)	2	2	1	2	40	2	2	1	2	300	4	2	1	2	850
8		(5,7,6)	2	3	1	3	50	3	3	1	3	280	4	3	1	3	880
9		(8,4)	3	2	2	2	60	3	2	2	2	320	5	2	2	2	900

Note: I: inspection time; CO: fixture cool time; m: Number of assembly lines; f: types of fixtures; n_f: number of copies of each type of fixture; q: product types; T: time window.

UPMSP has certain similarity with the proposed HALSFC problem (Lee et al., 2013; Sels et al., 2015). We select TS as a competitor to verify the performance of the proposed algorithm. The TS has the same neighborhood structure as IALNS, but it employs a distinct strategy for conducting the neighborhood search. Moreover, it is well known that genetic algorithm (GA) is also one of the mainstream metaheuristics. The algorithm is frequently employed to tackle NP-hard problems and has demonstrated exceptional performance in addressing the parallel machine scheduling problem (Cochran et al., 2003; Chaudhry and Drake, 2009). To better evaluate the performance of the proposed IALNS, we select GA as another competitor. We consider order-based crossover and two points crossover when designing GA operators, which refer to the study of Sivrikaya-Şerifolu and Ulusoy (1999).

All the model and algorithms are implemented by Python programming language and executed on a personal computer with 9th Gen Intel Core, 2.60 GHz processor and 16GB of RAM. The MILP is solved by Gurobi 10.0.1 with time limit of 1800s.

5.2. Computational results

In this section, we compare the computational results of MILP, IALNS, TS and GA on different-scale of instances. The gap for IALNS, TS and GA is defined by equation 32 and

the gap of MILP is obtained from Gurobi solver.

$$Gap = \frac{Obj_{MILP} - Obj_{Alg}}{Obj_{MILP}} * 100\%; \quad Alg = IALNS, TS, GA \tag{32}$$

Table 7 shows the computational results of small-scale instances with different scenarios. It is evident that MILP, IALNS and GA outperform TS in terms of objective value while all of the former methods achieve the same result. From the perspective of running time, the state-of-the-art exact technique, MILP, solved by Gurobi, outperforms IALNS, TS and GA in most cases, with the exception of scenario 4 and 6. The difference between IALNS, TS and GA in solving efficiency is not significant, with all methods exhibiting superiority over the others in certain examples. This observation demonstrates the applicability of Gurobi in solving optimization problems with small-scale instances, aligning with our shared understanding.

Table 8 shows the computational results of medium-scale instances. We can see that IALNS obtains competitive results compared to MILP, TS and GA in terms of both solution efficiency and solution quality. Due to the complexity of HALSFC, MILP even fails to find a feasible solution within time limit for some instances. With the increase in instance scale, this phenomenon becomes increasingly obvious. As table 9 shows, more large instances are unsolved for MILP within the time limit. Compared to MILP, IALNS, TS and GA achieve better results due to the their heuristic framework, which is suitable to tackle large instances. Specifically, IALNS outperforms TS in both solution efficiency and solution quality. Although the average running time of GA is shorter, the quality of the solution obtained by GA is not as good as that of IALNS in most cases. The above results demonstrate the advantages of IALNS in effectively solving HALSFC, particularly dealing with large-scale problems.

Table 7: Comparison of computational results by MILP, Algorithms IALNS, TS and GA with n = 20

Scenario	Instance	MILP			Alg IALNS			Alg TS			Alg GA		
		Obj	CPU(s)	Gap(%)	Obj	CPU(s)	Gap(%)	Obj	CPU(s)	Gap(%)	Obj	CPU(s)	Gap(%)
1	J101	14	13.42	0	14	29.45	0	13	26.60	7.14	14	30.68	0
	J102	14	8.96	0	14	30.09	0	14	33.08	0	14	30.32	0
	J103	14	3.84	0	14	34.09	0	13	33.57	7.14	14	46.45	0
Avg value		14	8.74	0	14	31.21	0	13.33	31.08	4.76	14	35.82	0
2	J104	14	0.96	0	14	28.57	0	14	28.71	0	14	42.36	0
	J105	13	23.8	0	13	26.63	0	13	24.02	0	13	32.19	0
	J106	14	0.96	0	14	75.45	0	12	36.08	14.29	14	27.09	0
Avg value		13.67	8.57	0	13.67	43.55	0	13.00	29.60	4.76	13.67	33.88	0
3	J107	13	0.44	0	13	30.50	0	12	30.44	7.69	13	35.64	0
	J108	14	1.21	0	14	34.97	0	14	28.82	0	14	39.36	0
	J109	14	1.25	0	14	31.95	0	14	29.92	0	14	32.56	0
Avg value		13.67	0.97	0	13.67	32.47	0	13.33	29.73	2.56	13.67	35.85	0
4	O101	16	49.67	0	16	35.60	0	13	32.60	18.75	16	21.48	0
	O102	14	21.97	0	14	38.46	0	14	29.71	0	14	32.15	0
	O103	17	52.31	0	17	38.53	0	13	30.41	23.53	17	33.04	0
Avg value		15.67	41.32	0	15.67	37.53	0	13.33	30.91	14.09	15.67	28.89	0
5	O104	12	20.23	0	12	39.24	0	12	27.39	0	12	43.82	0
	O105	13	27.52	0	13	34.07	0	13	37.43	0	13	37.64	0
	O106	13	23.10	0	13	30.00	0	12	27.67	7.69	13	35.93	0
Avg value		12.67	23.62	0	12.67	34.43	0	12.33	30.83	2.56	12.67	39.13	0
6	O107	13	22.16	0	13	30.21	0	13	33.77	7.69	72	38.94	0
	O108	15	37.59	0	15	30.04	0	14	28.29	6.67	15	28.19	0
	O109	13	59.75	0	13	31.70	0	13	29.45	0	13	49.56	0
Avg value		13.67	39.83	0	13.67	30.65	0	13.00	30.50	4.79	13.67	38.90	0
7	M101	13	1.15	0	13	77.89	0	12	35.67	7.69	13	33.07	0
	M102	15	1.59	0	15	29.92	0	12	30.18	20.00	15	38.63	0
	M103	14	4.43	0	14	36.91	0	14	30.99	0.00	14	31.58	0
Avg value		14.00	2.39	0	14.00	48.24	0	12.67	32.28	9.23	14.00	34.43	0
8	M104	14	13.16	0	14	29.62	0	13	31.95	7.14	14	44.97	0
	M105	13	6.36	0	13	26.69	0	13	33.71	0	13	30.64	0
	M106	14	1.69	0	14	35.63	0	12	29.52	14.29	14	50.43	0
Avg value		13.67	7.07	0	13.67	30.65	0	12.67	31.73	7.14	13.67	42.01	0
9	M107	13	18.06	0	13	29.06	0	12	32.58	7.69	13	35.09	0
	M108	14	0.60	0	14	40.20	0	13	35.08	0.07	14	23.63	0
	M109	14	1.74	0	14	30.31	0	13	32.67	0.07	14	46.53	0
Avg value		13.67	6.80	0	13.67	33.19	0	12.67	33.44	2.61	13.67	35.08	0
Total Avg value		13.86	15.48	0	13.86	35.84	0	12.93	31.06	5.84	13.86	36.00	0

Note: $Instance$: the example number; Obj: the objective function value; CPU: the computer running time; ——: for these instances, the feasible solutions cannot be obtained within 1800 seconds.

Table 8: Comparison of computational results by MILP, Algorithms IALNS, TS and GA with n = 100

Scenario	Instance	MILP			Alg IALNS			Alg TS			Alg GA		
		Obj	CPU(s)	Gap(%)	Obj	CPU(s)	Gap(%)	Obj	CPU(s)	Gap(%)	Obj	CPU(s)	Gap(%)
1	J201	86	55.83	0	73	160.46	15.12	74	160.13	13.95	78	195.55	9.30
	J202	74	1800.00	2.98	72	157.31	2.70	73	174.58	1.35	74	177.75	0.00
	J203	83	1800.00	4.55	70	157.45	15.66	72	177.40	13.25	83	184.28	0.00
Avg value		81.00	1218.61	2.51	71.67	158.41	10.72	73.00	170.70	9.52	78.33	185.86	3.10
2	J204	–	–	–	73	156.62	–	73	171.65	–	71	198.88	–
	J205	83	1800.00	4.83	72	161.06	13.25	70	272.56	15.66	63	191.09	24.10
	J206	79	1800.00	3.46	69	170.07	12.66	70	158.93	11.39	66	170.18	16.46
Avg value		–	–	–	71.33	162.58	–	71.00	201.05	–	66.67	186.71	–
3	J207	–	–	–	71	160.89	–	72	161.14	–	73	188.57	–
	J208	86	54.83	0	74	155.56	13.95	73	179.96	15.12	70	173.34	18.60
	J209	71	1800.00	9.52	73	162.12	-2.82	77	151.51	-8.45	66	196.20	7.04
Avg value		–	–	–	72.67	159.52	–	74.00	164.20	–	69.67	186.04	–
4	O201	70	617.09	0	58	158.44	17.14	74	162.87	-5.71	60	199.59	14.29
	O202	74	552.08	0	56	150.96	24.32	70	161.26	5.41	62	169.72	16.22
	O203	69	1161.39	0	59	159.46	14.49	58	160.18	15.94	66	196.20	7.04
Avg value		71.00	776.85	0	57.67	156.29	18.65	67.33	161.44	5.21	58.00	187.06	18.38
5	O204	80	1800.00	1.29	57	157.18	28.75	63	163.72	21.25	63	200.47	21.25
	O205	–	–	–	62	159.02	–	58	176.70	–	60	189.17	–
	O206	78	1572.5	0	57	155.53	26.92	61	158.49	21.79	60	198.84	23.08
Avg value		–	–	–	58.67	157.24	–	60.67	166.30	–	61.00	196.16	–
6	O207	63	1800.00	3.35	58	158.12	7.94	57	174.67	9.52	57	189.62	9.52
	O208	69	529.27	0	60	153.05	13.04	63	153.49	8.70	55	199.73	20.29
	O209	67	328.06	0	56	158.92	16.42	60	161.93	10.45	58	162.36	13.43
Avg value		66.33	885.78	1.12	58.00	156.70	12.47	60.00	163.36	9.56	56.67	183.90	14.42
7	M201	68	1607.48	4.42	66	154.08	2.94	55	167.99	19.12	62	161.72	8.82
	M202	75	328.43	0	70	161.43	6.67	61	155.48	18.67	66	185.40	12.00
	M203	71	1800.00	3.53	72	169.62	-1.41	57	183.45	19.72	60	178.03	15.49
Avg value		71.33	1245.30	2.65	70.33	161.71	2.73	57.67	168.97	19.17	62.67	175.05	12.11
8	M204	71	390.53	0	70	161.01	1.41	52	188.96	26.76	62	193.82	12.68
	M205	73	1800.00	6.61	73	166.71	0.00	55	179.85	24.66	63	181.40	13.70
	M206	71	1800.00	4.31	74	164.78	-4.23	47	175.70	33.80	70	163.87	1.41
Avg value		71.67	1330.2	3.64	73.00	164.17	-0.94	51.33	181.50	28.41	65.00	179.70	9.26
9	M207	65	1800.00	6.13	70	165.53	-7.69	60	212.24	7.69	68	193.07	-4.62
	M208	72	1800.00	5.26	77	164.27	-6.94	52	177.40	27.78	61	185.05	15.28
	M209	63	497.85	0	63	163.02	0	63	158.62	0	63	160.28	0.00
Avg value		66.67	1365.95	3.80	73.33	164.27	-4.88	58.33	182.75	11.82	64.00	179.46	3.55
Total Avg value		–	–	–	67.24	159.98	–	63.86	173.10	–	64.67	184.44	–

Note: *Instance*:the example number; *Obj*: the objective function value; *CPU*:the computer running time; ——: for these instances, the feasible solutions cannot be obtained within 1800 seconds.

Table 9: Comparison of computational results by MILP, Algorithms IALNS, TS and GA with n = 300

Scenario	Instance	MILP			Alg IALNS			Alg TS			Alg GA		
		Obj	CPU(s)	Gap(%)	Obj	CPU(s)	Gap(%)	Obj	CPU(s)	Gap(%)	Obj	CPU(s)	Gap(%)
1	J301	–	–	–	224	485.68	–	210	537.64	–	220	394.78	–
	J302	248	831.40	0	221	476.37	10.89	204	590.44	17.74	230	456.74	7.26
	J303	226	1800.00	6.51	224	488.78	0.88	212	639.65	6.19	231	396.94	-2.21
Avg value		–	–	–	223	483.61	–	208.67	589.24	–	227.00	416.15	–
2	J304	231	1800.00	2.29	226	485.30	2.16	204	597.05	11.69	214	413.60	7.3
	J305	–	–	–	229	488.52	–	216	625.32	–	227	456.28	–
	J306	220	1800.00	5.82	226	484.78	-2.73	212	594.67	3.64	224	479.83	-1.82
Avg value		–	–	–	227.00	486.20	–	210.67	605.68	–	221.67	449.90	–
3	J307	235	1800.00	3.11	227	468.67	3.40	208	571.38	11.49	216	419.12	8.09
	J308	237	5.44	2.27	226	492.09	4.64	205	486.50	13.50	216	409.65	8.86
	J309	–	–	–	231	486.77	–	210	542.70	–	241	416.45	–
Avg value		–	–	–	228.00	482.51	–	207.67	533.53	–	224.33	415.08	–
4	0301	231	1652.26	0	222	482.43	3.90	162	485.08	29.87	169	431.23	26.84
	0302	–	–	–	231	506.19	–	158	534.89	–	179	493.61	–
	0303	172	1800.00	12.24	189	460.19	-9.88	163	411.00	5.23	179	493.32	-4.07
Avg value		–	–	–	214.00	482.94	–	161.00	476.99	–	175.67	472.72	–
5	0304	–	–	–	178	455.86	–	167	434.06	–	166	456.79	–
	0305	–	–	–	178	449.80	–	156	479.35	–	162	477.06	–
	0306	210	583.02	0	177	460.59	15.71	163	439.19	22.38	161	461.14	23.33
Avg value		–	–	–	177.67	455.42	–	162	450.87	–	163.00	396.29	–
6	0307	–	–	–	178	457.11	–	161	466.44	–	173	392.30	–
	0308	232	1800.00	3.24	180	451.89	22.41	154	437.02	33.62	192	486.12	17.24
	0309	230	1800.00	5.83	179	469.21	22.17	157	446.69	31.74	161	468.24	30.00
Avg value		–	–	–	179.00	459.40	–	157.33	450.05	–	175.33	448.88	–
7	M301	–	–	–	215	433.87	–	107	454.29	–	203	390.73	–
	M302	203	1800.00	6.47	208	448.91	-2.46	156	455.40	23.15	189	500.43	6.90
	M303	–	–	–	211	460.18		127	459.78	–	202	495.29	–
Avg value		–	–	–	211.33	447.66	–	130.00	456.49	–	198.00	462.15	–
8	M304	186	90.53	3.65	195	448.76	-4.84	155	382.79	16.67	178	465.80	4.30
	M305	–	–	–	200	494.41	–	143	401.06	–	200	432.39	–
	M306	190	12.57	5.78	193	497.34	-1.58	105	450.41	44.74	115	483.46	39.47
Avg value		–	–	–	196.00	480.17	–	134.33	411.42	–	164.33	460.55	–
9	M307	–	–	–	189	503.70	–	118	462.69	–	144	429.31	–
	M308	180	1800.00	7.56	183	490.25	-1.67	122	461.26	32.22	168	471.02	6.67
	M309	169	1800.00	11.28	176	489.21	-4.14	120	449.95	28.99	165	412.83	2.37
Avg value		–	–	–	182.67	494.39	–	120.00	457.97	–	159	450.89	–
Total Avg value		–	–	–	204.30	474.14	–	167.05	493.46	–	189.81	444.91	–

Note: $Instance$: the example number; Obj: the objective function value; CPU:the computer running time; ——: for these instances, the feasible solutions cannot be obtained within 1800 seconds.

Table 10: Comparison of average computational results by MILP, Algorithms IALNS, TS and GA in different scale of instances

Instance scale		MILP	IALNS	TS	GA
small	Obj	13.86	13.86	12.93	13.86
	$CPU(s)$	15.48	35.84	31.06	36.00
	$Gap(\%)$	0	0	5.84	0
medium	Obj	–	67.24	63.86	64.67
	$CPU(s)$	–	159.98	173.10	184.44
	$Gap(\%)$	–	–	–	–
large	Obj	–	204.30	167.05	189.81
	$CPU(s)$	–	474.14	493.46	444.91
	$Gap(\%)$	–	–	–	–

Note: Obj: the objective function value; $CPU(s)$: the computer running time; —: for these instances, the optimal solutions cannot be obtained within 1800 seconds.

The summarized experimental results for three scales of instances are presented in Table 10. For small-scale instances, both IALNS and GA achieve the same solution quality as MILP, while TS has an average gap of 5.84%. For medium-scale instances, certain instances pose challenges for MILP to obtain feasible solutions within the specified time limit, resulting in unattainable average objective function values and other metrics. In such cases, IALNS exhibits a slightly slower solution speed compared to TS and GA but delivers higher-quality solutions. For large-scale instances, almost half of the instances cannot be solved by MILP while IALNS outperforms TS and GA in terms of both solution quality and computational efficiency. These findings highlight the computing advantages of IALNS for HALSFC problem under large-scale instances.

5.3. Sensitivity analysis

Considering the variability in experimental outcomes resulting from different parameter settings, this section investigates the impact of parameter setting. Sensitivity analysis of inspection time, the proportion of related work orders and the minimum cooling time of fixtures on the objective function and gap is presented in Figure 6 - Figure 11, respectively.

In order to investigate the impact of variations in quality inspection time on experimental results, we conduct five sets of numerical experiments at a consistent scale, solely altering the

quality inspection time I. We refer to the parameter settings of the medium-scale calculation example in Table 8. As the processing time p_{ij} follows $(5, 30)$ uniform distribution, we set the range of interval inspection time I to 0 to 20 during sensitivity analysis. From the experimental results shown in the Figure 6–7, we can see that with the increase of interval inspection time I, the objective value of each iteration has a trend of decreasing, and the final solution obtained is also smaller. Meanwhile, it is evident that a decrease in the quality inspection time leads to faster algorithm convergence. This implies that a shorter interval inspection time has minimal impact on the overall model operation and does not disrupt the operational process. However, once the quality inspection time surpasses a certain threshold, both the algorithm's solution speed and final outcome deteriorate, specifically, fewer work orders are completed within the specified timeframe.

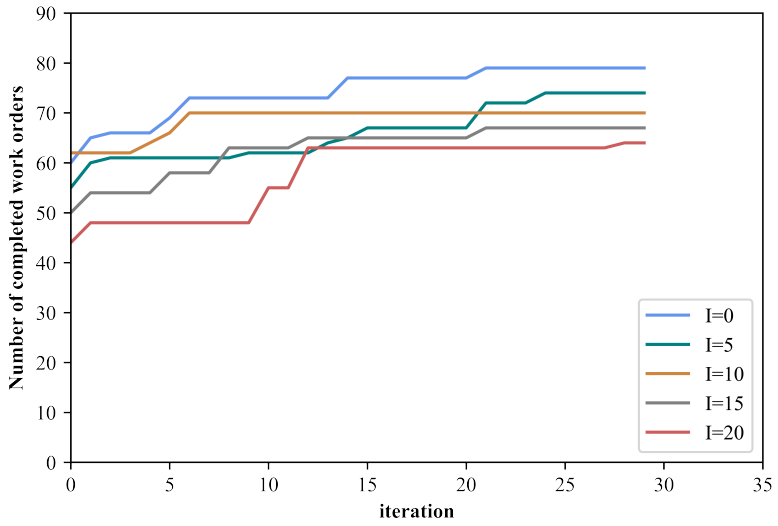

Figure 6: Influence of quality inspection time on objective function value.

In Section 5, we define the fixed proportion of related work orders as 30%. However, it is important to note that the proportion of related work orders may significantly impact the final solution. Therefore, we conduct a sensitivity analysis by gradually varying the proportion from 0% to 40%, in order to investigate its influence on the experimental results. To ensure consistency, we also refer to the numerical experiment in Section 5.2 and maintain identical parameter settings for sensitivity analysis.

Based on the experimental results depicted in Figure 8–9, it is evident that as the proportion of related work orders increases, the objective function value decreases during each iteration, leading to a smaller final solution. This phenomenon can be attributed to the influence of related work orders on the original scheduling strategy. When there are fewer related

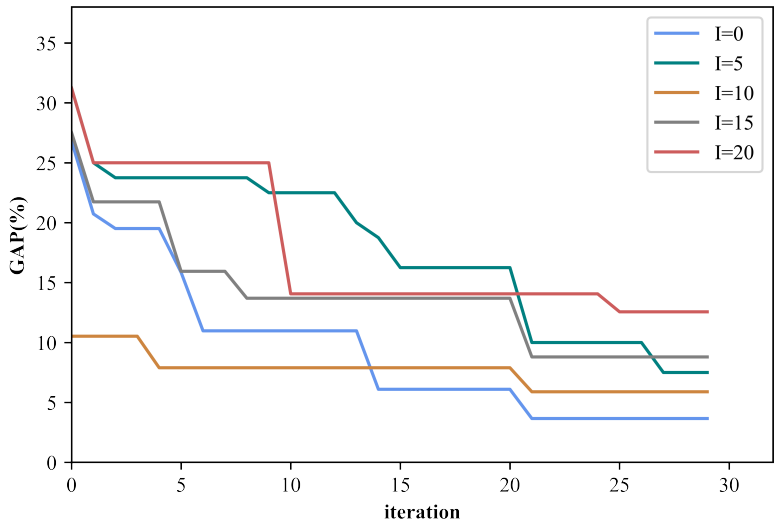

Figure 7: Influence of quality inspection time on GAP(%).

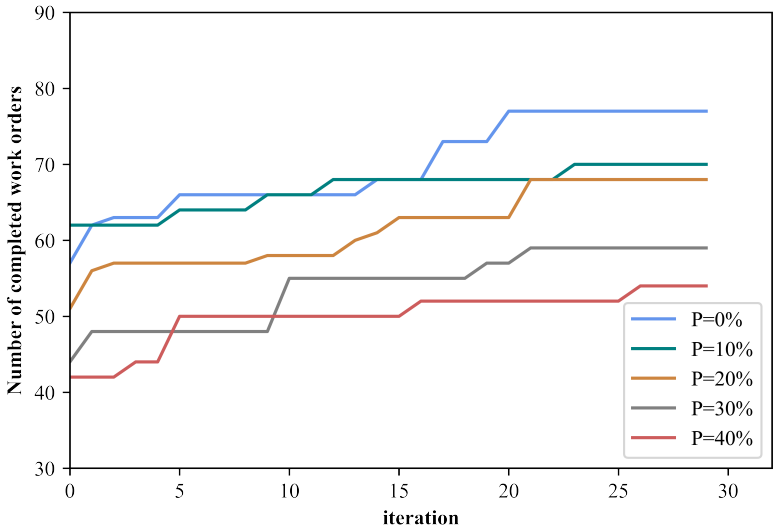

Figure 8: Influence of related work orders proportion on objective value.

work orders, the algorithm tends to prioritize other work orders more extensively, resulting in improved solution quality. However, when numerous related work orders exist, the algorithm must first schedule those with larger quantities before addressing those with smaller quantities. Consequently, scheduling outcomes with a higher number of related work orders tend to be inferior.

In addition to the parameter settings involved in the related work orders, we further investigate the influence of the fixture setting. We set the cooling time for each fixture to be the same to observe the change of solution quality and efficiency. Therefore, we conduct a sen-

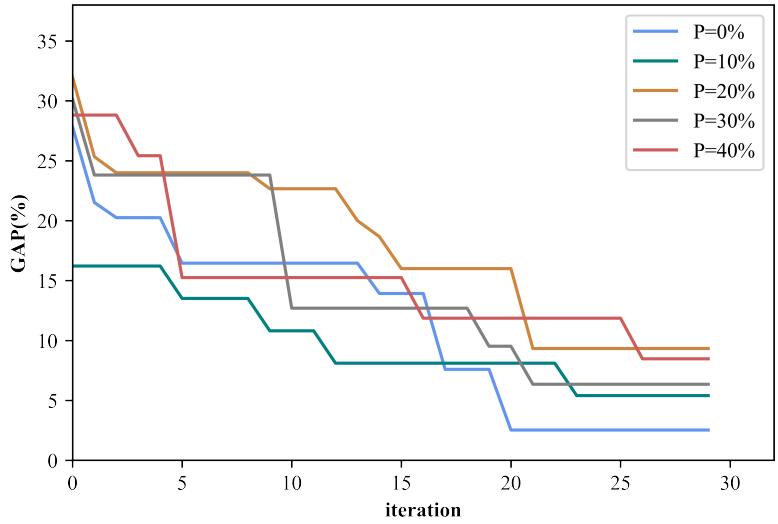

Figure 9: Influence of related work orders proportion on GAP(%).

sitivity analysis by gradually varying the minimum fixture cooling time (CO) from 5s to 25s. Without changing other parameter settings, we use the instances in scenario 5 for experiments.

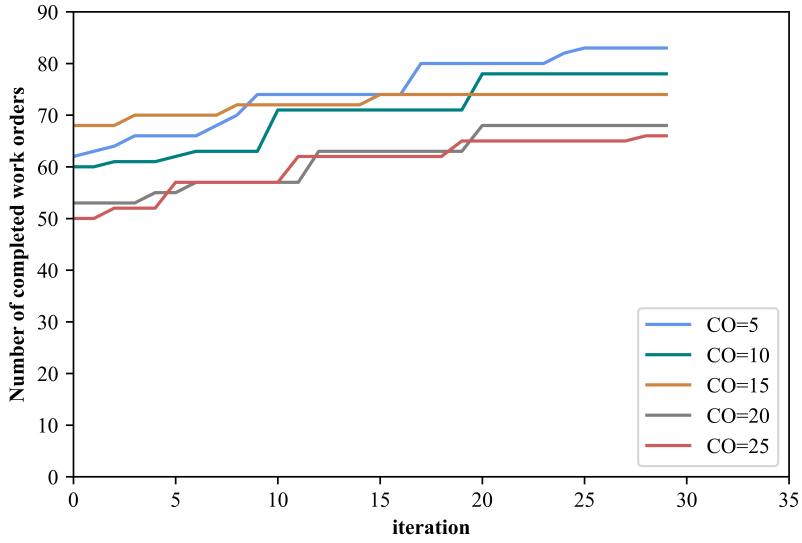

Figure 10: Influence of minimum fixture cooling time on objective value.

As shown in Figure 10–11, as the minimum fixture cooling time increases, the objective function value decreases. This phenomenon may be attributed to that the increase of the minimum cooling time reduces the time that the fixtures can be utilized for processing product in the fixed time T , which further reduces the number of completed work orders. It can be seen that the selection of fixtures should prioritize those with a lower minimum cooling time

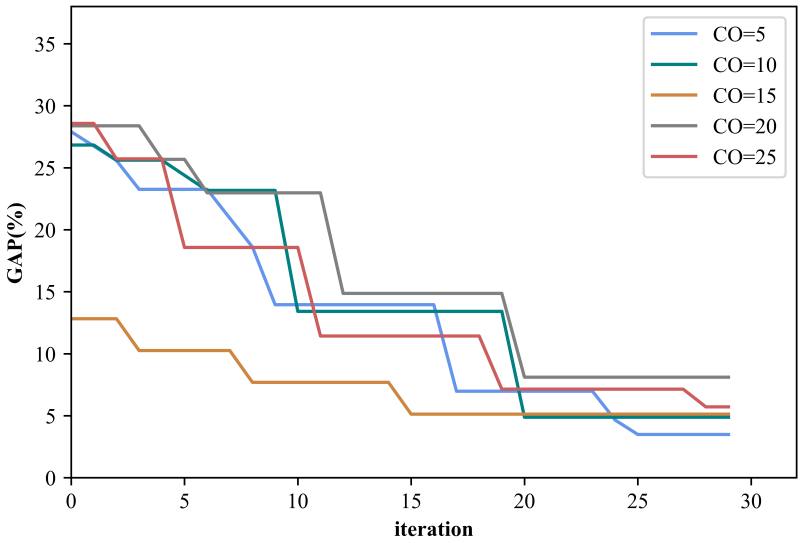

Figure 11: Influence of minimum fixture cooling time on GAP(%).

when multiple fixtures share the same other parameters. Alternatively, accelerating the cooling method to reduce the fixture rest time can also enhance production efficiency.

6. Conclusions and future research

In this study, inspired by the real production scenario of a intelligent terminal manufacturing enterprise, we address a parallel heterogeneous assembly line scheduling problem with fixture constraints. The HALSFC problem contains consecutive processing time constraints, fixture cooling time constraints, fixture quantity constraints, and related work order constraints, aiming to maximize the completed work orders in a given time horizon. We formulate a mixed integer linear programming model for HALSFC and propose a lower bound and a upper bound for the objective. To effectively solve this complex problem, we develop an improved adaptive large neighborhood search algorithm, hybridizing with the framework of simulated algorithm. Among them, the minimum product quantity of the work orders first allocating strategy (MF) is proposed to improve the quality of the initial solution. The experimental results show the effectiveness of MF strategy in solving HALSFC compared with other common strategies, e.g., FCFS, EDD, SPT, RD. Four destroy operators and two repair operators are designed to generate the neighborhood, and metropolis acceptance criterion and adaptive weight strategy are utilized. Extensive experiments confirm the effectiveness and efficiency of the proposed method. Additionally, sensitivity analysis on the proportion of related work orders, the quality inspection time and the minimum fixture cooling time are conducted. The results demonstrate that

an increase in the interval inspection time, the proportion of related work orders and the minimum fixture cooling time leads to a degradation in solution quality. This phenomenon can be attributed to the increased duration of interval inspections, the corresponding proportion of work orders and minimum fixture cooling time, resulting in more intricate constraints that limit the scheduling effectiveness.

Fixtures are an essential component in the actual production process of many manufacturing enterprises. Despite this, many enterprises still rely on manual methods to select and schedule fixtures, which significantly hampers production efficiency. This study proposes an optimization method for the digital transformation of manufacturing enterprises. Intelligent algorithms, such as IALNS, GA, and TS, have the potential to enhance enterprise decision-making by reducing reliance on human involvement, improving decision-making capabilities, and increasing cost-effectiveness and efficiency.

For future research, the proposed model can be enhanced by integrating additional constraints from real production scenarios, thereby increasing the realism of the MILP model. Furthermore, incorporating alternative acceleration strategies for IALNS can further accelerate the algorithm. Moreover, exploration of exact algorithms for HALSFC is also a potential direction.

References

Abu-Marrul, V., Martinelli, R., Hamacher, S., and Gribkovskaia, I. (2021). Matheuristics for a parallel machine scheduling problem with non-anticipatory family setup times: Application in the offshore oil and gas industry. *Computers & Operations Research*, 128:105162.

Afzalirad, M. and Rezaeian, J. (2016). Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. *Computers & Industrial Engineering*, 98:40–52.

Akbar, M. M., Manning, E. G., Shoja, G. C., and Khan, S. (2001). Heuristic solutions for the multiple-choice multi-dimension knapsack problem. In *Computational Science-ICCS 2001: International Conference San Francisco, CA, USA, May 28—30, 2001 Proceedings, Part II 1*, pages 659–668. Springer.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European journal of operational research*, 246(2):345–378.

Allahverdi, A. (2016). A survey of scheduling problems with no-wait in process. *European Journal of Operational Research*, 255(3):665–686.

Bennage, WA. and Dhingra, AK. (1995). Single and multiobjective structural optimization in discrete-continuous variables using simulated annealing. *International Journal for Numerical Methods in Engineering*, 38(16):2753–2773.

Bitar, A., Dauzère-Pérès, S., and Yugma, C. (2021). Unrelated parallel machine scheduling with new criteria: Complexity and models. *Computers & Operations Research*, 132:105291.

Boysen, N., Schulze, P., and Scholl, A. (2022). Assembly line balancing: What happened in the last fifteen years? *European Journal of Operational Research*, 301(3):797–814.

Bressanin, J. M., Guimarães, H. R., Chagas, M. F., de Mesquita Sampaio, I. L., Klein, B. C., Watanabe, M. D. B., Bonomi, A., de Morais, E. R., and Cavalett, O. (2021). Advanced technologies for electricity production in the sugarcane value chain are a strategic option in a carbon reward policy context. *Energy Policy*, 159:112637.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., and Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European journal of operational research*, 112(1):3–41.

Chaudhry, I. A. and Drake, P. R. (2009). Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms. *The International Journal of Advanced Manufacturing Technology*, 42:581–594.

Chen, J.-F. and Wu, T.-H. (2006). Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. *Omega*, 34(1):81–89.

Chen, M., Liu, Q., Huang, S., and Dang, C. (2022). Environmental cost control system of manufacturing enterprises using artificial intelligence based on value chain of circular economy. *Enterprise Information Systems*, 16(8-9):1856422.

Cheng, TCE. and Lin, BMT. (2017). Demonstrating Johnson's algorithm via resource-constrained scheduling. *International Journal of Production Research*, 55(11):3326–3330.

Cochran, J. K., Horng, S.-M., and Fowler, J. W. (2003). A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers & Operations Research*, 30(7):1087–1102.

Da Silva, B. J. V., Morabito, R., Yamashita, D. S., and Yanasse, H. H. (2014). Production scheduling of assembly fixtures in the aeronautical industry. *Computers & Industrial Engineering*, 67:195–203.

Dell'Amico, M., Iori, M., Martello, S., and Monaci, M. (2008). Heuristic and exact algorithms for the identical parallel machine scheduling problem. *INFORMS Journal on Computing*, 20(3):333–344.

Fang, K., Uhan, N. A., Zhao, F., and Sutherland, J. W. (2013). Flow shop scheduling with peak power consumption constraints. *Annals of Operations Research*, 206:115–145.

Fang, K., Wang, S., Pinedo, M. L., Chen, L., and Chu, F. (2021). A combinatorial Benders decomposition algorithm for parallel machine scheduling with working-time restrictions. *European Journal of Operational Research*, 291(1):128–146.

Fang, K.-T. and Lin, B. M. (2013). Parallel-machine scheduling to minimize tardiness penalty and power cost. *Computers & Industrial Engineering*, 64(1):224–234.

Fanjul-Peyro, L., Perea, F., and Ruiz, R. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *European Journal of Operational Research*, 260(2):482–493.

Fanjul-Peyro, L., Perea, F., and Ruiz, R. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *European Journal of Operational Research*, 260(2):482–493.

Fanjul-Peyro, L., Ruiz, R., and Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research*, 101:173–182.

Fleszar, K. and Hindi, K. S. (2018). Algorithms for the unrelated parallel machine scheduling problem with a resource constraint. *European Journal of Operational Research*, 271(3):839–848.

França, P. M., Gendreau, M., Laporte, G., and Müller, F. M. (1994). A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective. *Computers & operations research*, 21(2):205–210.

Glass, C. A., Potts, CN., and Shade, P. (1994). Unrelated parallel machine scheduling using local search. *Mathematical and Computer Modelling*, 20(2):41–52.

Gökgür, B., Hnich, B., and Özpeynirci, S. (2018). Parallel machine scheduling with tool loading: A constraint programming approach. *International Journal of Production Research*, 56(16):5541–5557.

Gunawan, A., Widjaja, A. T., Vansteenwegen, P., and Vincent, F. Y. (2020). Adaptive large neighborhood search for vehicle routing problem with cross-docking. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Hashemian, N., Diallo, C., and Vizvári, B. (2014). Makespan minimization for parallel machines scheduling with multiple availability constraints. *Annals of Operations Research*, 213:173–186.

He, L., Guijt, A., de Weerdt, M., Xing, L., and Yorke-Smith, N. (2019). Order acceptance and scheduling with sequence-dependent setup times: A new memetic algorithm and benchmark of the state of the art. *Computers & Industrial Engineering*, 138:106102.

Herr, O. and Goel, A. (2016). Minimising total tardiness for a single machine scheduling problem with family setups and resource constraints. *European Journal of Operational Research*, 248(1):123–135.

Herroelen, W., De Reyck, B., and Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research*, 25(4):279–302.

Herroelen, W. S. (1972). Resource-constrained project scheduling—the state of the art. *Journal of the Operational Research Society*, 23(3):261–275.

Hunter, R., Rios, J., Perez, JM., and Vizan, A. (2006). A functional approach for the formalization of the fixture design process. *International Journal of machine tools and manufacture*, 46(6):683–697.

Jiang, E.-d. and Wang, L. (2019). An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time. *International Journal of Production Research*, 57(6):1756–1771.

Kiefer, A., Hartl, R. F., and Schnell, A. (2017). Adaptive large neighborhood search for the curriculum-based course timetabling problem. *Annals of Operations Research*, 252:255–282.

Kim, D.-W., Kim, K.-H., Jang, W., and Chen, F. F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, 18(3-4):223–231.

Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., and Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial intelligence review*, 13:129–170.

Lee, J.-H., Yu, J.-M., and Lee, D.-H. (2013). A tabu search algorithm for unrelated parallel machine scheduling with sequence-and machine-dependent setups: Minimizing total tardiness. *The International Journal of Advanced Manufacturing Technology*, 69:2081–2089.

Lee, K. and Chuah, K. (2001). A super methodology for business process improvement-an industrial case study in hong kong/china. *International Journal of Operations & Production Management*, 21(5/6):687–706.

Lei, D., Yuan, Y., and Cai, J. (2021). An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling. *International Journal of Production Research*, 59(17):5259–5271.

Lin, Y.-K. and Hsieh, F.-Y. (2014). Unrelated parallel machine scheduling with setup times and ready times. *International Journal of Production Research*, 52(4):1200–1214.

Liu, X., Laporte, G., Chen, Y., and He, R. (2017). An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. *Computers & Operations Research*, 86:41–53.

Lopez-Esteve, A., Perea, F., and Yepes-Borrero, J. C. (2023). Grasp algorithms for the unrelated parallel machines scheduling problem with additional resources during processing and setups. *International Journal of Production Research*, 61(17):6013–6029.

Ma, W., Che, Y., Huang, H., and Ke, H. (2016). Resource-constrained project scheduling problem with uncertain durations and renewable resources. *International journal of machine learning and cybernetics*, 7:613–621.

Maecker, S., Shen, L., and Mönch, L. (2023). Unrelated parallel machine scheduling with eligibility constraints and delivery times to minimize total weighted tardiness. *Computers & Operations Research*, 149:105999.

Mara, S. T. W., Norcahyo, R., Jodiawan, P., Lusiantoro, L., and Rifai, A. P. (2022). A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, 146:105903.

Mokotoff, E. (2004). An exact algorithm for the identical parallel machine scheduling problem. *European Journal of Operational Research*, 152(3):758–769.

Mokotoff, E. and Chrétienne, P. (2002). A cutting plane algorithm for the unrelated parallel machine scheduling problem. *European Journal of Operational Research*, 141(3):515–525.

Ozbakir, L., Baykasoglu, A., Gorkemli, B., and Gorkemli, L. (2011). Multiple-colony ant algorithm for parallel assembly line balancing problem. *Applied Soft Computing*, 11(3):3186–3198.

Pellerin, R., Perrier, N., and Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 280(2):395–416.

Peng, J. and Liu, B. (2004). Parallel machine scheduling models with fuzzy processing times. *Information Sciences*, 166(1-4):49–66.

Pfeiffer, C. and Schulz, A. (2022). An ALNS algorithm for the static dial-a-ride problem with ride and waiting time minimization. *Or Spectrum*, 44(1):87–119.

Pinto, G., Ben-Dov, Y. T., and Rabinowitz, G. (2013). Formulating and solving a multi-mode resource-collaboration and constrained scheduling problem (MRCCSP). *Annals of Operations Research*, 206:311–339.

Prilutskii, M. K. (2007). Multicriterial multi-index resource scheduling problems. *Journal of Computer and Systems Sciences International*, 46:78–82.

Priyadarshini, M., Das, I., Ghangrekar, M. M., and Blaney, L. (2022). Advanced oxidation processes: Performance, advantages, and scale-up of emerging technologies. *Journal of Environmental Management*, 316:115295.

Rajkanth, R., Rajendran, C., and Ziegler, H. (2017). Heuristics to minimize the completion time variance of jobs on a single machine and on identical parallel machines. *The International Journal of Advanced Manufacturing Technology*, 88:1923–1936.

Ranjbar, M. and Kianfar, F. (2010). Resource-constrained project scheduling problem with flexible work profiles: A genetic algorithm approach.

Recalde, D., Rutten, C., Schuurman, P., and Vredeveld, T. (2010). Local search performance guarantees for restricted related parallel machine scheduling. In *LATIN 2010: Theoretical Informatics: 9th Latin American Symposium, Oaxaca, Mexico, April 19-23, 2010. Proceedings 9*, pages 108–119. Springer.

Ribeiro, G. M. and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & operations research*, 39(3):728–735.

Ropke, S. and Pisinger, D. (2006a). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Ropke, S. and Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775.

Sánchez, M. G., Lalla-Ruiz, E., Gil, A. F., Castro, C., and Voß, S. (2022). Resource-constrained multi-project scheduling problem: A survey. *European Journal of Operational Research*.

Santini, A., Ropke, S., and Hvattum, L. M. (2018). A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, 24:783–815.

Santoro, M. C. and Junqueira, L. (2023). Unrelated parallel machine scheduling models with machine availability and eligibility constraints. *Computers & Industrial Engineering*, 179:109219.

Saraç, T. and Tutumlu, B. (2022). A bi-objective mathematical model for an unrelated parallel machine scheduling problem with job-splitting. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 37(4):2293–2308.

Sels, V., Coelho, J., Dias, A. M., and Vanhoucke, M. (2015). Hybrid tabu search and a truncated branch-and-bound for the unrelated parallel machine scheduling problem. *Computers & Operations Research*, 53:107–117.

Sivrikaya-Şerifoğlu, F. and Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research*, 26(8):773–787.

Sivrikaya-Şerifolu, F. and Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research*, 26(8):773–787.

Tao, Z. and Hao, C. (2009). Dynamic classified JSP scheduling based on petri net and GASA. In *2009 IEEE International Conference on Automation and Logistics*, pages 532–537. IEEE.

Tian, J., Fu, R., and Yuan, J. (2014). Online over time scheduling on parallel-batch machines: A survey. *Journal of the Operations Research Society of China*, 2(4):445–454.

Vallada, E. and Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 211(3):612–622.

Wang, H. and Alidaee, B. (2019). Effective heuristic for large-scale unrelated parallel machines scheduling problems. *Omega*, 83:261–274.

Wang, J., Song, G., Liang, Z., Demeulemeester, E., Hu, X., and Liu, J. (2023a). Unrelated parallel machine scheduling with multiple time windows: An application to earth observation satellite scheduling. *Computers & Operations Research*, 149:106010.

Wang, P., Reinelt, G., and Tan, Y. (2012). Self-adaptive large neighborhood search algorithm for parallel machine scheduling problems. *Journal of Systems Engineering and Electronics*, 23(2):208–215.

Wang, S., Wu, R., Chu, F., and Yu, J. (2022). Unrelated parallel machine scheduling problem with special controllable processing times and setups. *Computers & Operations Research*, 148:105990.

Wang, Y., Che, A., and Feng, J. (2023b). Energy-efficient unrelated parallel machine scheduling with general position-based deterioration. *International Journal of Production Research*, 61(17):5886–5900.

Wang, Y.-C., Wang, M.-J., and Lin, S.-C. (2017). Selection of cutting conditions for power constrained parallel machine scheduling. *Robotics and Computer-Integrated Manufacturing*, 43:105–110.

Wauters, T., Verbeeck, K., Verstraete, P., Berghe, G. V., and De Causmaecker, P. (2012). Real-world production scheduling for the food industry: An integrated approach. *Engineering Applications of Artificial Intelligence*, 25(2):222–228.

Wen, M., Linde, E., Ropke, S., Mirchandani, P., and Larsen, A. (2016). An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. *Computers & Operations Research*, 76:73–83.

Weng, M. X., Lu, J., and Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International journal of production economics*, 70(3):215–226.

Xiao, S., Wu, Z., Sun, S., and Jin, M. (2021a). Research on the dual-resource constrained robust job shop scheduling problems. *J. Mech. Eng*, 57:227–239.

Xiao, Y., Zheng, Y., Yu, Y., Zhang, L., Lin, X., and Li, B. (2021b). A branch and bound algorithm for a parallel machine scheduling problem in green manufacturing industry considering time cost and power consumption. *Journal of Cleaner Production*, 320:128867.

Xiufan, Z. and Decheng, F. (2023). Collaborative emission reduction research on dual-pilot policies of the low-carbon city and smart city from the perspective of multiple innovations. *Urban Climate*, 47:101364.

Yalaoui, F. and Chu, C. (2002). Parallel machine scheduling to minimize total tardiness. *International journal of production economics*, 76(3):265–279.

Yepes-Borrero, J. C., Perea, F., Ruiz, R., and Villa, F. (2021). Bi-objective parallel machine scheduling with additional resources during setups. *European Journal of Operational Research*, 292(2):443–455.

Yunusoglu, P. and Topaloglu Yildiz, S. (2022). Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, 60(7):2212–2229.

Zeng, J. and Zhang, X. (2021). An adaptive large neighborhood search for single-machine batch processing scheduling with 2-D rectangular bin-packing constraints. *IEEE Transactions on Reliability*, 71(1):139–148.

Zhigang, L., Yan, L., and Shujuan, L. (2007). Multi-resource constrained job-shop optimization scheduling based on ant colony algorithm. *Journal of System Simulation*, 19(1):216–220.

Zhou, J., Li, P., Zhou, Y., Wang, B., Zang, J., and Meng, L. (2018). Toward new-generation intelligent manufacturing. *Engineering*, 4(1):11–20.

Zhou, J., Love, P. E., Wang, X., Teo, K. L., and Irani, Z. (2013). A review of methods and algorithms for optimizing construction scheduling. *Journal of the Operational Research Society*, 64:1091–1105.